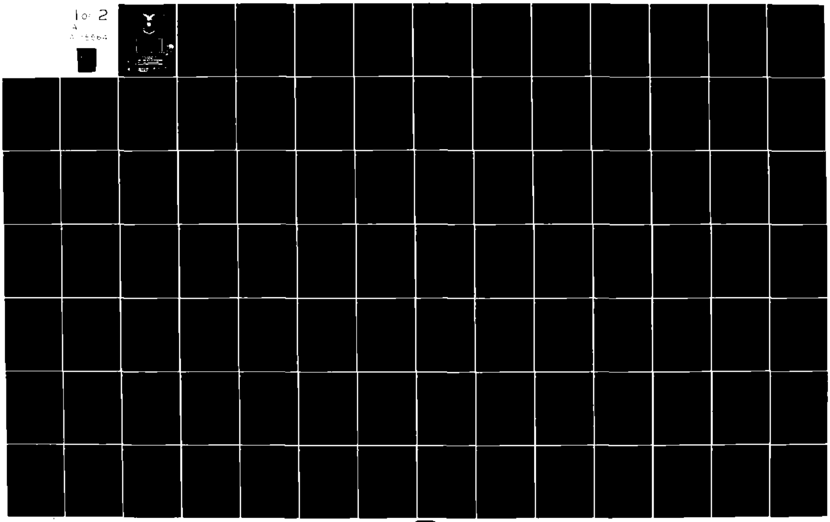
AD-A115 564 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOO--ETC F/G 9/2
MICROPROGRAMMING: A TOOL TO IMPROVE PROGRAM PERFORMANCE.(U)
DEC 81 J J STEIDLE
UNCLASSIFIED AFIT/GE/EE/81D-56 NL

1 of 2



AD A115564

MICROPROGRAMMING: A TOOL TO IMPROVE

PROGRAM PERFORMANCE

THESIS

AFIT/GE/EE/81D-56 JOHN J. STEIDLE

CIVILIAN EMPLOYEE USAF

AFIT/GE/EE/81D-56

MICROPROGRAMMING: A Tool to Improve Program Performance

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

by

John J. Steidle

Civilian Employee, USAF

Graduate Electrical Engineering

December 1981

CONTENTS

PAGE

Acknowledgements iv

List of Figures v

List of Tables vi

Abstract vii

I. Introduction 1

Background 1

Problem 4

Scope 4

Approach 5

Sequence of Presentation 6

II. Microprogramming Concept 8

Introduction 8

How Microprogramming Provides a Speed Gain 8

Writable Control Store as a System Resource 10

Advantages and Disadvantages of HP Microprogramming 12

Analyzing Where to Microprogram 15

Activity Profile Concept 17

Summary 18

III. Electronic Warfare Computer Program Survey 19

Introduction 19

Programs Surveyed 21

General Criteria for Selection 21

Activity Profile Experiements 22

Implementing an Activity Profile Generator 24

Results 24

CONTENTS (Cont'd)

		Page
	Conclusions	25
	Summary	27
IV.	Implementing a Microprogramming Capability	28
	Introduction	28
	Literature and HP Documentation Survey	28
	Base Survey	29
	HP 21MX Computer (M-Series)	32
	RTE-III Microprogramming System Generation	32
	Microprogramming Software Support	36
	Problems Encountered	37
	Results	38
	Summary	38
V.	A Microprogrammed Example: Fast Fourier Transform	39
	Introduction	39
	Fast Fourier Transform Test Program	39
	Method	40
	FFT Profile Analysis	42
	Predicted Improvement via Microprogram Enhancement	45
	Microprogram Requirements	45
	Design of the Microcode	47
	Bit Reversal Algorithm	49
	Microprogramming the Bit Reversal	50
	Testing the Microprogram	50
	Effectiveness of the Microprogrammed Bit Reversal	53
	Summary	62

CONTENTS (Cont'd)

	Page
VI. Results and Conclusions	63
Introduction	63
Results	63
Conclusions	64
Recommendations	65
Bibliography	68
Appendix A: List of Programs Surveyed	70
Appendix B: Activity Profile Generator Program Listing	76
Appendix C: List of Facilities Visited	81
Appendix D: 21MX M-Series User Microprogrammable Computer	82
Appendix E: RTE-III System Generation: ANSWER File Listing	88
Appendix F: List of Commands Used for Backing Up System on Disk	108
Appendix G: Discussion of Implementation Problems Encountered During Study	110
Appendix H: How to Run ACTV	112
Appendix I: Sample Profile Analysis	114
Appendix J: Listing of Microprograms and Assembly Interface Routine Developed	124
Appendix K: Theoretical Computation of the Run Time For Microprogrammed Bit Reversal Sorting Routine	132
VITA	134

ACKNOWLEDGEMENTS

I am deeply indebted to Mr. John Bankovskis of the Air Force Avionics Lab for his expert assistnace and technical knowledge, concerning the HP21MX computer system and HP RTE-III operating system, without which this project could not have been completed on time.

Gratitude is also expressed to Mr. Jim Leonard of the Air Force Avionics Lab for his invaluable guidance, technical discussions and use of his activity profile program and HP-21MX computer system to check out the microprograms developed on the AFIT computer.

I wish to thank my thesis adviser, Dr. Gary Lamont for his valuable assistance in this effort. I also wish to thank the members of my thesis committee, Capt. Larry Kizer and Maj. Charles Lillie for their valuable comments while reviewing the thesis drafts.

Special thanks is given to Ms. Doreen Dixon and Ms. Charlene Shaw for their expert typing efforts.

Finally, I wish to thank my wife, June for her patience and understanding during my AFIT assignment and especially during those final months when this report was being written.

List of Figures

Figure		Page
1	Block Diagram of AFIT 21MX and EWAF 21MX Computer System	31
2	Activity Profile Summary for FOUR1 and FOURE FFT Programs	43
3	Flow Chart of Bit Reversal Sorting Routine	48
4	Microprogram to Form Bit Reversed Number	51
5	Comparison of Run Time for FOUR1 Program	54
6	Comparison of Run Time for FOURE Program	55
7	Comparison of Run Time for FOUR1 Program with Microprogram	57
8	Comparison of Run Time for FOURE Program with Microprogram	58
9	Comparison of Run Time for Bit Reversal Code	60
10	Comparison of Speed Up Factor for Microprogram	61
11	21MX Computer Architecture	83
12	FOUR1 Activity Profile Plot	120
13	FOURE Activity Profile Plot	123
14	Flow Diagram of Bit Reversal Sorting Routine	131

List Of Tables

Table		Page
I	HP Hardware Available for Project	30
II	AFIT RTE-III Microprogramming System Configuration ..	34
III	AFIT RTE-III System Assignments and Driver Layout Used	35
IV	Comparison of Max. Differences for FOURE FFT Subroutine	44
V	Relative % Improvement Factor Over Software FFT Routine	56
VI	Address Load Map For FOUR1 and FOURE FFT Programs ...	117
VII	Activity Profile Results for FOUR1 FFT Program	118
VIII	Activity Profile Results for FOURE FFT Program	121

Abstract

A user microprogramming capability was implemented on AFIT's HP 21MX RTE-III computer system. The AFIT computer will be used to support student research in microprogramming projects involving real time digital signal processing and special time critical programs for military environments.

This study further looks at a specific microprogramming technique to tailor application programs for improving a programs execution time. A feasibility study to analyze program activity for microprogram improvements is done for the fast Fourier transform. The Bit Reversal sorting routine is microcoded to demonstrate the technique. Response of the FFT programs is analyzed using an activity profile generator program, and the difference in execution speed of the programs with and without the microprogrammed bit reversal routine is measured and compared.

MICROPROGRAMMING: A TOOL TO IMPROVE PROGRAM PERFORMANCE

I. Introduction

The topic is motivated primarily by the desire to promote use of a powerful tool for tailoring application programs such as those employed in digital signal processing, post-data reduction or computer graphics to meet special requirements. For example, real time execution, accuracy or special wordlength requirements are often required when interfacing hardware equipment to the computer. This study however deals primarily with using microprogramming to improve a programs execution time, whether a real time application or other. This chapter presents the objectives and background requirements, problem and scope for undertaking this investigation.

Background

A significant number of laboratory facilities at Wright-Patterson AFB, including the Air Force Institute of Technology (AFIT) use HP-21MX computer systems for a variety of post-data processing analysis and real time data acquisition, digital signal and control tasks. The Electronic Warfare Analysis Facility (EWAF), managed by the Electronic Warfare Division ASD/ENAD, is one such facility which uses a 21MX (2112) computer system to support a wide range of SPO related electronic warfare (EW) tests, evaluation and analysis tasks. Present EWAF applications include in-house and contracted tasks for the preparation, acquisition, presentation and post-processing of EW simulation test data and

antenna measurement data. Future applications include real time digital signal waveform processing tasks and program simulations for EW analog models of various threat receivers for analyzing response to jamming waveforms.

One unique feature the 21MX system supports is a user dynamic microprogramming capability (Ref. 1:3-15). This feature gives a user the flexibility to augment, modify or completely replace the computers basic machine instruction set or to program special purpose algorithms to meet specific application needs. For example in real time digital signal processing work, throughput and execution speed are generally of prime importance. By using appropriate microprogramming techniques, execution speeds can be significantly improved. In other applications, such as data reduction tasks, special instructions could be microcoded to improve accuracy beyond that which the base computer can provide and at the same time improve throughput performance.

Unfortunately microprogramming is somewhat specialized and complex. The technique is still viewed as a difficult science by many computer users who have not been sufficiently exposed to microprogramming concepts to understand and put to use its' many benefits (Ref. 2, Ref. 3:210). And most end users do not realize its ability to speed up execution of time-critical routines.

Microprogramming concepts were first introduced over 30 years ago by M. V. Wilkes (Ref. 4) but applications are still in an early stage of development (Ref. 1:3-19). Wilkes' application of microprogramming delt primarily with hardware for designing the control processor unit (CPU) (Ref. 1:1). Today however, with the advant of user microprogram-

mable computers, fast semi-conductor memories, and cheap microprogrammable bit-slice microcomputer chips (Ref. 5:98), interest in using microprogramming techniques in non-cpu applications is rising rapidly (Ref. 6:33). The availability of the HP-21MX user microprogrammable computer makes it practical to use and explore microprogramming techniques for a wide range of applications.

Microprogramming in general is more difficult and time consuming then writing programs in FORTRAN or ASSEMBLY languages and requires the programmer to know more about the specific internal timing and architecture of the computer (Ref. 7:33). The HP microprogramming language however is very similar to writing programs in ASSEMBLY language and is only slightly more difficult to learn. A relatively simple mnumonic language is used by HP and the resulting code assembled using a microassembler. Microprogram instructions are stored in a special high speed random access memory called a Writable Control Store (WCS).

Because microprogramming is more specialized, complex and time consuming, HP end users may shy away from developing specific applications which could be of benefit. AFIT could provide a significant focal point for HP users at Wright-Patterson AFB and elsewhere by experimentation through AFIT research projects involving microprogramming applications. For example real time input-output problems, digital signal processing applications and specialized tuning of user programs could be studied to enhance execution speed, accuracy, and word length considerations. The benefits include extending the useful life of HP equipment at the various labs at WPAFB, developing more usage and better understanding for a very powerful computer application tool, and student

training in microprogramming techniques and computer control designs. Additionally firmware concepts being used in Air Force avionics systems could be effectively demonstrated to students resulting in a greater awareness and appreciation for the technology.

Problem

The problem this investigation addresses is implementing a system to support dynamic user microprogramming. Two basic resources are required to undertake this study. The first is a user microprogrammable computer. The second is a reasonable operating system which supports microprogramming. The first requirement is met with Hewlett-Packard's 21MX computer. The second requirement is met with Hewlett-Packard's Real-Time Executive (RTE) operating system. Neither the RTE operating system or the microprogramming capability has been implemented on AFIT's HP 21 (2108) computer equipment. In addition, this study investigates a specific non-cpu application of microprogramming relating to EW simulation and data processing programs. The conceptual use of HP microprogramming to tailor digital signal processing and post data processing application programs for improving execution response is studied. The concept of using microprograms to tailor the machine or application programs to meet specific applications is not new (Ref. 7). Part of this latter problem consist of identifying appropriate means and general criteria to be applied for selection of application programs which show promise of being improved by microprogramming techniques.

Scope

The development of the microprogramming capability making up this study includes the following:

1. The installation and checkout of a 256 word writable control store (WCS) memory module.
2. RTE-III system generation to include microprogramming.
3. Implementation and checkout of various HP software support tools necessary for microprogram development in an RTE operating environment.
4. An RTE-III backup generation on disk.
5. Identification of HP decumentation and survey of microprogramming differences between HP 2100 and HP 21MX computers (M, E, F series).
6. A base survey of HP users.
7. A literature search.
8. Activity analysis of FFT.
9. Development of a microcoded bit reversal sorting routine.

Approach

A limited survey study was made of the available signal processing, post data processing and related EW evaluation programs being supported now or planned in the future by the EWAF facility. From this survey several programs were chosen and examined more closely to determine where and how microprogramming could be used to an advantage to improve and tune execution performance. An activity profile generator routine was used to monitor the software activity for program analysis.

A detailed profile analysis for two similar FORTRAN FFT programs was done using the activity profile generator program. A bit reversal sorting algorithm for the [illegible] [illegible]. This micro-

program was incorporated as a subroutine call in the FFT programs to demonstrate the feasiblity of improving program execution time using microprograms. No special attempt was made to implement the fastest or most accurate microcode. The programs were tested and the improvement in execution time determined and compared with that predicted from the activity profile analysis.

The limited time available for this project did not permit explicit investigation of some important tradeoff options such as program accuracy, word length considerations and interrupt status. The reader should be aware, however, such tradeoff variables are extremely important considerations for designing microprograms to optimize execution response, especially for time critical applications. For example, checking interrupt status is a primary responsibility the microprogrammer must consider when writing a microprogram (Ref. 8:7-21). Interrupts, of course, affect the speed of the microprogram. If the interrupt rate is too high the speed benefits offered by microprogramming may be lost.

Sequence of Presentation

This report documents the results of an investigation conducted during the spring and summer quarters of 1981 at AFIT to implement a microprogramming capability on AFIT's HP 21MX computer. Chapter I has presented the objectives and background requirements for undertaking the investigation. In order to put this report into context Chapter II presents a brief overview of the microprogramming concept, writable control store and the activity profile concept. Chapter III discusses the survey of the electronic warfare programs examined,

presents the general criteria used to select several programs for analysis with an activity profile generator and presents the analysis results which led to the selection of the two FFT programs for demonstrating the microprogramming capability. However before any microprogramming could actually be done in this study it was first necessary to put together AFIT's 21MX computer system. Chapter IV briefly discusses the implementation phase, the system generation required and the HP microprogramming support software available. Then in Chapter V the design and test of an FFT bit reversal algorithm in microcode is considered. Finally in Chapter VI the results and conclusions of the study are presented and recommendations for further development are made.

II. Microprogramming Concept

Introduction

Microprogramming, in general, is a very powerful and useful tool. This chapter establishes that microprogramming has the unique capability to significantly reduce execution time for a computer program. Although other uses of microprogramming exist, the first section briefly explains how microprograms provide a speed gain to decrease overall software program execution time. Microprograms usually reside in a special memory in the computer control section called a writable control store. The second section emphasizes some of the important properties of the writable control store as a system resource. The next section higlights some of the advantages and disadvantages of microprogramming and its application to computers in general as well as the HP 21MX computer. To apply microprogramming techniques effectively, one needs to know where and what to microprogram. The fourth section discusses one useful tool, the activity profile, for determining where to apply microprogram techniques. The last section covers the activity profile concept and how to generate a profile.

How Microprogramming Provides a Speed Gain

In general, microprogramming provides a speed gain from the following five sources:

1. Elimination of instruction fetches.
2. Faster memory components.
3. Extra storage registers.
4. Parallel processing.
5. More ALU operations available.

With microprogramming a user can eliminate memory fetch instructions in a software program by combining a series of machine instructions into one single microprogram. Usually in a software program each machine instruction requires a fetch from main memory. In contrast, only one fetch from main memory is required to execute the single microprogram. Since a memory fetch takes approximately 35% to 45% of the CPU time (Ref 2:11), the overall software program execution time is decreased in direct proportion to the number of machine instructions replaced. Thus a significant time savings can be realized by this factor alone.

The other factors listed above also contribute to additional increases in speed. Microinstructions reside in a special high speed memory in the control store section of the computer. The control store memory may be either Read Only Memory (ROM) or special Writable Control Store (WCS) memory. It is important to note that this control store memory is independent of the main memory. The cycle time required to access the control memory is 2 to 5 times faster than for main memory. User microprograms can reside in either WCS or be "burned" into non-volatile PROM chips which can be installed on User Control Store (UCS) boards. The microprogrammed computer instruction set for the HP computer for example is burned into a set of ROM chips and reside on a special board mounted below the motherboard. As a result of using faster memory components to store microprogram instructions, efficiency is increased 2 to 5 times that of calling individual machine instructions according to Hewlett Packard.

In addition to the control store being more than twice as fast as

main memory, microinstructions have access to many internal high speed scratch pad registers that main memory machine instructions cannot use. What this means is that having more scratch pad registers will result in fewer main memory fetches and thus achieve faster execution (Ref 9:8).

Each microinstruction word in the HP 21MX is 24 bits wide verses 16 bits for the software machine instruction. The 24 bit words are divided into fields which directly define the control steps to be executed within the system. With the 21MX and other microprogrammable machines each field performs a different micro operation, often independently of one another. Consequently many different operations can be performed in parallel to maximize efficiency. This is in contrast to the machine instruction which can usually only perform a single operation. It is important to observe, however, that each operation performed by a machine instruction is actually a series of microinstructions which defines the operation. The 16 bits making up the machine instruction in fact defines the starting location of a microprogram which performs the machine operation.

Finally more arithmetic and logical functions are available on the microcode level which main memory programs cannot use. This makes it possible to take full advantage of the machines architecture, internal registers and logical features to more precisely match the implementation of a program operation or algorithm. Consequently program efficiency can be optimized.

Writable Control Store as a System Resource

In the previous section it was explained that microprogramming

achieves a speed gain through the use of faster memory components, i.e. either ROM or Writable Control Store modules. Because "... many users do not fully understand what WCS is, what it can do, and most important, what it means to them." (Ref 10:16) as a user, this section will highlight some of the properties of the WCS as a system resource.

First, WCS is simply a high speed memory device. It is independent of the main computer memory system. It is used to hold user microinstructions. In contrast, macroinstructions (machine language instructions) always reside in main memory. Usually the WCS word is much wider than a main memory instruction word.

Second WCS is an I/O device. What this means is the computer can perform input and output operations to the WCS device like any other I/O device. This permits storing user microprograms in main memory or on disk files and later loading them into the WCS as required. This makes it possible to alter the computer operations dynamically as a main memory program executes if a WCS option is installed.

The WCS board is simple to install in the computer. It fits into an I/O slot provided in the printed circuit assembly (PCA) cage of the computer.

Third, one important factor which must be weighted befored using a WCS for program production or control tasks is the fact that the WCS memory is volatile. If power fails in the computer the instructions in WCS will be lost. This could result in a dangerous situation if the computer is controlling a hazardous operation. Consequently this factor of volatility must be weighed with the application when deciding whether to use ROM or WCS.

Fourth, adding more WCS will improve the computers capability and

flexibility. WCS is available for the HP system in various sizes either a 256 word board or a lk word board. Several boards may be installed in the computer system. This means longer and more complex programs can be accommodated but also implies longer execution of a microprogram will result. This will make it necessary then to consider the problem of program interrupts when designing long microprograms. Fortunately most microprograms are relatively short.

The writable control store is the key to user microprogram development in the HP 21MX RTE operating system. With the WCS option installed the user can load, execute and debug his microprograms on-line while operating with RTE system. The WCS also allows improved interaction with the RTE editor, disk file manipulations and so on for easier microprogram development (Ref 11:15).

Further, in HPs real time multiprogramming environment many uers can access a single set of microprograms resident in WCS or multiple user programs can use the same WCS area and sequentually load in different microprograms. And finally, if more than one WCS module is available each module can be used independently. This permits a structure where a combination of separate and identical microprograms can be accessed by multiple users (Ref 11:16).

It should be pointed out however that the sharing of WCS can pose some major coordination problems (Ref 12:21).

Advantages and Disadvantages of Microprogramming

Some of the advantages offered by microprogramming are as follows:

1. Speed enhancement
2. Flexibility
3. Tailorability

4. Vertical microinstructions allow easy microprogramming

5. Instruction enhancement

6. Special instructions

7. More control than assembly language

8. Excellent software support tools offered by HP

9. Dynamic user microprogramming capability

The speed enhancement which microprogramming provides has been discussed. But microprogramming offers more than just a speed advantage as seen from the list above. Microprogramming also permits a great flexibility in both the design of computers and in the definition of the computers base instruction set. With a microprogrammed computer this flexibility permits the freedom to tailor the insturction set for special instructions or to enhance existing instructions as new or better algorithms become available. In addition the HP 21MX computer is user microprogrammable which means anyone can use this feature; it is not just limited to the vendor. But more importantly, the HP microprogramming language uses a vertical or diagonal (Ref 13:14) microinstructions. This type of microinstruction is simpler to use and makes the HP microprogramming language relatively easy. To aid user microprogramming development, various software support tools are uaually necessary. Hewlett Packard solves this by providing a powerful debug/editor program.

Some of the disadvantages offered by microprogramming are as follows:

Time to write microprograms - not for day to day programming

Documentation and maintenance may be difficult

Easy to crash system - best to debug on a dedicated system

Software not machine transportable

Verification procedure is difficult

The above disadvantages of microprogramming are mostly self-explanatory. Instead of stepping forward to an easier, higher ordered and more structured language, the user is actually taking a step in the opposite direction to an almost unstructured language in which many events can take place in parallel and at random times. This makes microprogramming orders of magnitude more difficult to write, document and maintain.

Becuase microprograms control the basic machine, generally at the gate level, a great potential exists for making disastrous errors which could literally destroy any operating system, files and so forth on the machine. As a consequence, it is wise not to develop microprograms on a system which is running production jobs or controlling hazardous applications.

Usually microprogramming languages are computer specific, even among the same computer types. This means microprograms developed on one machine may not, necessarily, work on another. Likewise software programs that use the microprograms will not work on other machines. Because of the many interactions possible in microprogramming, verification of a microprogram is difficult. Until better tools are developed to test microprograms this will be an area which will hinder user microprogramming.

Some of the limitations of the HP 21MX are as follows:

1. Speed .... 320 ns cycle time
2. Read/Write .... Takes two cycles
3. WCS .... Limited size

In the 21MX M-series computer, the basic microinstruction time takes 325 ns to complete. This is the primary limiting factor on speed for real-time applications. Another limiting factor occurs when reading or writing from main memory. Two microinstruction cycles are required for either a read or write operation. Most microprograms tend to be short so a large WCS is not always required. However as applications grow, the size of microprograms will likely grow also. The M-series is limited by the WCS area it has available. About 2k words of WCS are available for user microprogramming in the M-series. This may not be sufficient area for large user microprograms especially those which use complex operations and floating point math.

Analyzing Where to Microprogram

Having identified the reasons why microprogramming can speed up a program and understanding what WCS is and can do for use we now turn to applying these methods.

Theoretically one would want to write an entire application program in microcode to achieve maximum performance. Such a task would be huge. Since the available WCS area in most microprogrammable computers is relatively small, the task would also be impractical.

An alternative solution is to microprogram portions of software which are frequently used. This is the microprogramming concept of tuning and involves analyzing the application program to determine areas which, if replaced with microcode, will result in significant savings of overhead. It is often stated that 90 percent of a program execution time is sent in 10 percent of the actual software code (Ref

2:7, Ref 14:83-87, Ref 15:56). If this is true then significant saving in execution time can result without expending a great deal of microprogramming time. It also imples microprograms will be short thus reducing the amount of effort spent in writing the code. But this leads to the real problem of identifying the software code which is limiting the programs performance.

How does one easily identify inefficient code, for example in large software programs? Determining what section of a program is actually inefficient is not always an obvious task (Ref 11:15). In the area of program timing quite often one is mislead by intution (Ref 12: 18-19). A portion believed to be taking the most execution time may not actually be doing so. Determiniation of what to microprogram in a specific application program is a function of many things. The process generally requires a critical analysis of the programs activity to determine inefficiencies accurately. For purposes of analysis several methods exist:

* Measure system time using a watch or the computers builtin clock to time the various sections.
* Use a special hardware tool known as a breakpoint register such as the HP 10676A.
* Generate an activity profile of the program.

The first method is simple but may require considerable insight into what to measure since all time information is reduced to a single number.

The second method is extremely useful and accurate but requires special hardware be used (Ref 12:99).

The third method is the most useful. An activity profile is simply a table of frequency counts of a computers program address counter (P-register). This data is plotted to show the percentage of time a program spends in each area of memory. Snook in his paper (Ref 14:33-57) remarks that a profile is useful for several reasons. One is to locate the areas where a program spends most of its time. Two, it identifies those areas which use very little time (therefore improving these areas is usually a waste of time). Finally, one gains a better insight into the program being used.

Snook, in addition, points out the profile generator helps reduce the amount of guess work involved in optimizing a program by pinpointing the bottleneck areas which may require a simple recoding or redesign of an algorithm to make it more efficient.

Activity Profile Concept

Several ways exist to generate an activity profile. The simplest method is to just halt the computer a number of times while the program is executing and tabulate the addresses of the p-register. The drawbacks to this method are obvious.

A second method is to use the computer itself to monitor and collect the execution activity of a program while it is running. For this purpose a simple program called an ACTIVITY PROFILE GENERATOR is often used. This is a software program which periodically interrupts the program being monitored and simply checks the p-register address and stores the results for latter analysis. Usually the address range of the program can be specified which also allows the user to zoom in on specific areas of the program which may be of high interest.

When the users program ends the activity profile data can then be analyzed by means of a histogram plot or cummulative probability plot of the data. By comparing the output results with the loader address map for the program one can readily isolate areas of high and low activity in the program. As a result, the profile analysis will indicate what sections of source code can be modified, reprogrammed in assembly language or reprogrammed in microcode to improve execution response time.

Summary

This section introduced the concept of using microprogramming as a useful tool for improving the execution speed of a program. How microprogramming achieves a speed gain was covered and some advantages and disadvatages of microprogramming were outlined. The activity profile concept is a useful tool for analyzing a program. It can be used to advantage especially for finding out the inefficient software portions of program. Once the inefficient software, is identified, it can be microprogrammed resulting in an improvement in overall execution time for the program.

III. ELECTRONIC WARFARE COMPUTER PROGRAM SURVEY

Introduction

Before any microprogramming effort could be undertaken for this study it was first necessary to survey the spectrum of EW programs available on the EWAF facility. The approach taken was to compile a list of the available candidate EW simulation models, post-data processing and digital signal processing programs being planned or currently operating on the EWAF facilities computer. Then from this list chose one or more programs for a more detailed analysis to determine those areas that could be microprogrammed for improved performance.

As discussed earlier the EWAF is a multi-user computer facility used primarily to support a wide range of SPO related EW evaluation and analysis tasks. In general the following areas are supported:

- EW radar/missile simulation models
- EW test data preparation and post-data processing
- Statistical and regression analysis programs
- Real time signal processing and data acquisition tasks

At the present time the real time signal processing and data acquisition capability is still in a planning stage and is virtually non existant. Suitable programs in the following areas are desirable candidates for building up the EWAF real time signal processing capability:

A/D conversion	Logical decision making
Filter programs	Kalman filters
Integrators	Max. entropy filters
Correlators	Pattern recognition
Threshold detection	Fourier transforms
Signal conditioning	Walsh transforms

The following criteria was identified as a means of selecting several programs from the survey list for a more detailed analysis:

- Due to the limited time available for this study the programs selected for futher analysis were limited to FORTRAN programs currently running on the EWAF 21MX computer. As a result, many of the larger EW simulation programs, such as MPASS or MECCA, that currently run on the base CDC computer system were excluded.
- Improving the execution time of a program was the primary emphasis for using microprogramming in this study. It follows that programs which take excessive time to run on the EWAF should be considered prime candidates.
- Programs that can share common code algorithms, data structures, sorting routines etc. were considered next. Such programs would permit writing modular microprograms that can be used by other routines, requiring only a minimum of interfacing.
- Programs which are used often, especially in a multiprogramming environment were included. Frequently used programs should be microprogrammed. This would possibly permit speeding up the overall computer system response in a multiprogram environment as well as the individual program.
- The HP RTE operating system software and utility programs available on the EWAF were excluded from the survey; Source code for these programs was unavailable. Further special care in writing microprograms for the system software must be exercised so as not to void the vendors maintenance and [illegible].

- Any program selected should be reasonably simple to modify and interface to the microprograms. This criteria suggests looking at programs which show high clustered activity areas, so that microprograms can be written in modular forms. Programs having many subroutine blocks could be potential candidates.
- Any programs selected should have a simple set of input data available to check programs before and after microprogram improvement and to ease the burden of learning to run the program for this study.

Programs Surveyed

The list of candidate programs considered for this study is given in Appendix A. The actual FORTRAN coding for the programs are not presented in this report. Program details related to these programs are minimized for two reasons:

1. Comprehension of the activity profile results does not require an understanding of the program code.
2. It is beyond the scope of this effort to become an expert on the FORTRAN content of these programs.

From this list of programs the next problem was to select one or two for a detailed activity profile analysis.

General Criteria for Selection

The primary emphasis for improving program performance in this initial investigation is program execution speed. Other performance improvements using microcode, such as accuracy requirements, were discussed in Chapter II but are not considered here.

Activity Profile Experiments

Using the criteria above, activity profile experiments were run for a variety of programs selected from the survey list. The following programs were initially selected:

- BESSEL function program
- WEDGE program
- CROSS CORRELATION PROGRAM -- & CROSO
- AUTO CORRELATION & SPECTRIAL ANALYSIS -- & AUTSP
- STEPWISE REGRESSION -- & STEPR
- DOWNLINK JAMMING PROGRAMS
- FFT PROGRAMS

The BESSEL, STEPWISE REGRESSION, CROSS CORRELATION, and AUTO CORRELATION programs were selected on the bases of frequency of use and potential usefulness for improving the EW signal processing capability.

The DOWNLINK JAMMING and WEDGE programs were chosen because of execution time, frequency of use, and large number of subroutines used in the programs.

The FFT programs were selected for their commonality properties and their potential as basis of other digital signal processing programs such as power spectrum, computing ambiguity functions, and periodograms and real time applications.

An activity profile generator program (ACTV), discussed below, was used to obtain the profiles. For each program examined the following tasks were performed:

- Obtain the necessary documentation and input data, if available, for running the program.
- Compile each program to obtain a mixed FORTRAN/ASSEMBLY language listing.

- Load each program to get a listing of the loader address map.
- Run the ACTV program simultaneously with the program being monitored.
- Obtain a printout of the activity profile results and compare with the load map and FORTRAN listings of the monitored program to determine the software areas which could possibly be microprogrammed to decrease execution time.

Although the tasks described above are straight forward the time required to set up the programs, schedule computer time, obtain the necessary documentation and input data made it impractical to examine more of the surveyed programs.

A major set back occured after obtaining some initial profile results for these programs. The EWAF 21MX computer developed an unexplained problem which caused several disk files to be overwritten and also destroyed the RTE-IVB operating system. The activity profile generator program (ACTV) was one of the programs lost. In addition, the ACTV program was initially thought to have been the cause of the system crash since several files were overwritten with portions of the ACTV program. The ACTV program, however, was cleared and it was determined that the problem was actually in the EWAF hardware. Several weeks passed before the ACTV program was restored on the EWAF system.

At this point a decision was made to analyze the profiles already obtained and based on those results choose one program to examine in greater detail for a possible feasibility study.

Implementing an Activity Profile Generator

Implementing an activity profile generator program to analyze the selected programs presented some problems. First, it was discovered that the HP activity profile software available (Ref 16) was designed to operate with RTE III and would not run under the EWAF's RTE-IVB operating system. Further, it did not execute properly when tried on AFIT's RTE-III system. The program ran but the results were obviously wrong.

Hewlett-Packard sells an activity profile package (Ref 17) for the RTE-IVB system but this solution was both impractical and costly.

An alternative solution was found after talking with Mr. Jim Leonard (Ref 18) AFWAL/AARF. He wrote an activity profile generator for his RTE-IVB system and provided a copy of his profile (ACTV) program for use on the EWAF system. A listing of the activity profile (ACTV) program is provided in Appendix B. The ACTV program is similar to HP's version but is simplier to use and does not require any modifications be made to the program being monitored.

The activity profile program was successfully implemented on the EWAF computer but it would not execute on AFIT's RTE-III system because it lacked a system subroutine provided for in the RTE-IV system. It is believed it would not be difficult to write the required subroutine to make this program work on AFIT's computer. This was not pursued, however, at this time.

Results

The results of the profile analysis were unexpected as well as informative. First all the profiles examined exhibited a common characteristic. Each profile showed that the majority of activity in the programs examined occured in the RTE system library subroutines. This

overhead activity amounted to about 80% to 90% of the total activity of the programs examined. The major offending library routine was the system REIO routine. The REIO routine accounted for 20% to 40% of a programs total activity. The REIO routine is associated with swapping programs in memory and other I/O operations.

Second, the operations, such as complex multiplication and addition, are also system libraries. The profiles generated for a monitored program do not permit seperating out the activity in the complex routines and the software code which use them. However, in all cases examined, the activity in the complex routines amounted to less than 20% of the total activity. The activity generated by the REIO routine and I/O routines was significantly greater than the activity generated by the complex arithmetic operations.

Conclusions

The following conclusions were formed from this analysis. First, it was concluded it would be difficult to significantly decrease the overall execution time of the programs being studied without micro-coding the program system library routines. Microprogramming the system library routines however is beyond the scope of this study and would probably require extensive knowledge of the operating system to achieve effective results. This type of microprogramming effort might best be left to the vendor.

This exercise, however, clearly points out one major area where microprogramming a single system library subroutine such as REIO could perhaps speed up the entire operating system by as much as 20% to 40%. When considering a multiprogramming environment supporting many users such a speed up could easily justify the time spent in developing the microprogram.

Third, from the activity profiles generated it is clear that the responses of non-system subroutines in a program might be improved tremendously through the use of microprogramming but the resulting decrease in execution time may not have a great effect on the overall program performance. In contrast if a section of code such as a subroutine in a major software loop which is executed often the profile results would show a much greater activity in this section of code, and may indeed be the principle contributing factor to the overall program response time. Microcoding this type of activity would be very productive in terms of time spent in coding the routine as well as increasing the program speed.

Finally, a decision was made to examine the fast Fourier transform (FFT) programs in more detail based on the above finding and after a discussion with Mr. Phil Douville (Ref 19). Mr. Douville pointed out a program he was developing, which calls a 4098 point complex FFT routine 79 times and takes 2 hours to run on the 21MX computer. It can be seen that even a small speed up in the FFT routine could significantly reduce the time his program takes to run.

The FFT is a well known and frequently used program which has many applications in the EW analysis area as well as in many real time digital signal processing applications. A microprogramming FFT firmware option is available from Hewlett-Packard for their 21MX computer. However, attempting to analyze the FFT program activity and microprogram sections of the code has merit. Further it can adequately demonstrate the process and provide valuable user insight to microprogramming tuning concepts.

Summary

This chapter discussed the list of EW programs available and some general criteria form which several programs were chosen for further analysis using an activity profile generator program. The activity profile generator program, ACTV, was briefly discussed and the initial results of running profiles against several of the chosen programs resulted in an unexpected conclusion. The results indicated that the RTE system prgoram REIO was responsible for consumming a major portion of a programs time. It was concluded that the REIO utility routine would be an excellent program for microprogramming but because it was a system program was beyond the scope of this effort. Finally it was concluded that the FFT programs available offered the best candidate for demonstrating the microprogramming concepts.

IV. MICROPROGRAMMING IMPLEMENTATION PHASE

Introduction

A major challenge of this study was to assemble the various HP-21MX system components available at AFIT into a working system and to generate an RTE-III operating system that supports user microprogramming. To accomplish this integration, supplementary documentation on the HP-21MX computer equipment and operating system had to first be obtained. This was an on-going effort throughout the entire study. Part of this effort included a base survey of HP users to understand how others have set up and used their systems. Before any actual microprogramming effort could start it was also necessary to incorporate the HP microprogramming software support programs into the RTE-III system. Likewise, considerable effort was spent on learning the HP microprogramming language for the 21MX M-series computer and also understanding the differences in the language for the various other HP computer series. This chapter briefly highlights the 21 MX system that was assembled and the system generation performed.

Literature and HP Documentation Survey

Before undertaking any microprogramming effort a thorough understanding of the system architecture and the RTE-III operating system for the HP-21MX computer had to be achieved. This understanding was pursued during the entire investigation. It involved the collection and review of available HP documentation, a literature review

of professional publications and survey of facilities at WPAFB using HP computers.

A good portion of this effort involved the identification, cross reference, and functional understanding of the available HP system components for compatiblity with an RTE-III operating system. A list of the available equipment surveyed is included in Table I. The main result of this effort permitted assembly of the overall AFIT computer system for operation with an RTE-III operating system. Figure 1 shows a general block diagram of the AFIT system developed and the present EWAF system available at ASD/ENAD.

Base Survey

A base survey of HP computer users was undertaken. This survey primarily consisted of personal interviews with the various base users of HP computer equipment. The following objectives were accomplished by this effort:

- The applications to which HP computers were being put to use was determined.
- The extent to which HP microprogramming was being used was determined.
- A useful source of contacts experienced with using HP equipment was established.

A list of the various facilities visited is included in Appendix C.

The primary findings from this effort are as follows:

1. The microprogramming features of the 21MX were not used dispite the fact that several facilities had the necessary writable control store (WCS) option.

TABLE I

HP Hardware Available for Project

HP Product Option No.	Description
2108A	Computer ("A" power supply)
2102A	Memory Controllers
1202B	STD Performance Memory Controller
12747A	128kb. STD Performance Memory Module
12892B	Memory Project Module
12897B	Dual Channel Port Controller (DCPC)
12539C	Time Base Generator
12731A	Memory Expansion Module
12945A	User Control Store (USC)
12978A	Writable Control Store (WCS) 256 Words
12992B	Disk Loader ROM for 7905/7906/7920 Disk Unit
12992C	CRT Terminal Loader ROM for 2648A graphics terminal
12566B	Microcircuit duplex register modules
12555B	Dual 8-bit D/A converter module
2648A	Graphics Terminal with minicartridge I/O
13037-80023	Disk Interface Card
5060-6282	Papertape Photoreader Interface Card
12896-60001	DMA Board
12531	Buffered TTY Register
02100-60060	Terminator Boards
12998A	16k STD Performance Memory Modules
12994A	8k STD Performance Memory Modules

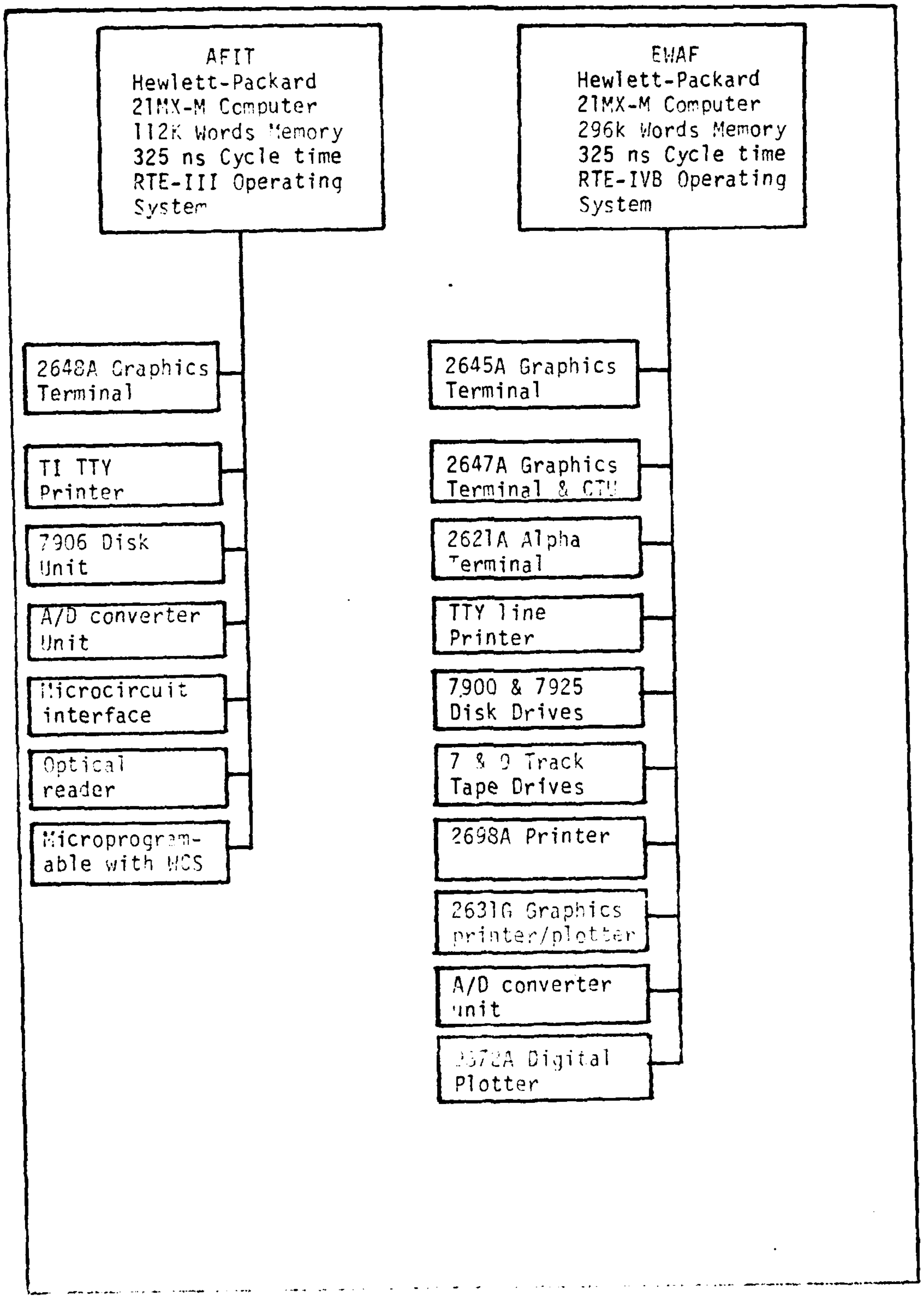


Figure 1. Block Diagram of AFIT 21MX and EWAF 21MX Computer System

2. None of the facilities were currently involved in using the HP systems for real time digital signal processing tasks.

3. Real time data acquisition and control applications were the primary tasks for which the HP systems were being used.

HP-21MX Computer (M-series)

Before studying microprogramming it is important to have a thorough understanding of the computers architecture and its various sections. This was achieved by studying the vendors documentation on the computer. A brief description of the 21MX computer characteristics and special features of microprogramming are presented in Appendix D. Further detailed information is available in HP service and operating manuals (Ref 20, Ref 21).

RTE-III Microprogramming System Generation

To configure a new operating system to support the HP microprogramming features required running the RTE-III on-line generator program RT3GN. A typical system generation normally takes several hours to run depending on the relocatable programs that are being added to make up the operating system. Unfortunately the RTE-III system generation proved to be more of a challenge than what it should have been. Ordinarily, the task of generating a new operating system is a relatively straight forward and simple procedure consisting of the following steps:

- plan your system
- generate an ANSWER file
- run the RTE-III on-line generator program - [illegible]

- run the RTE system transfer program - SWITCH
- perform a system backup.

Initial attempts to run the RT3GM program failed. It took several weeks of effort to finally get the program to successfully generate the new operating system. Each time the generator was run the program would execute part way through and then halt at random. The reason for the computer halting is unknown. An effort to find out why the halts occured, failed. Memory and system diagnostic tests were run but the tests were negative. Nothing was found to indicate anything was wrong. The author found however that cleaning the contacts of all the memory boards seemed to help reduce the incidence of the halts. Also rearranging the order of the relocatable programs in the ANSWER file seemed to help. It was later learned that other HP users on base have also had similar halting problems and have not found the basic cause. The typical solution has been to call in HP service and replace suspected memory boards with new ones. The major hardware and relocatable software modules which were configured into the system are listed in Table II. The various system assignments and driver layout used for this study in setting up the RTE-III microprogramming system configurations are listed in Table III. A listing of the ANSWER file used in this study for the microprogramming systems configuration is included in Appendix E. The actual details of generating a new RTE-III operating system are contained in Hewlett-Packard's RTE-II /RTE-III on-line generator reference manual (Ref 22).

Once the RT3GM program finally generated a new system configuration output file, the utility program SWTCH was used to transfer the new operating system created by the on-line generator program to a disk subchannel. See Section [illegible] of Ref 22 for detailed information about the use of SWTCH.

TABLE II

AFIT RTE-III Mcroprogramming System Configuration

HARDWARE MODULES	SOFTWARE MODULES
HP 2108 Computer	RTE-III Memory Resident System
112k Main Memory	RTE-III System Library
Memory Protect	RTE-III Compiler Library
Time Base Generator	Powerfail Driver, DVR 43
DCPC (Dual Cahnnel Port Controller)	EDITR (Interactive Editor)
Dynamic Maping System	RTE-III LOADR (Relocating Loader)
HP 7906 Disk Subsystem	MTM (Multi Terminal Generator)
HP 2548 Graphics System Console	RT3GN (RTE-III On-Line Generator)
TI TTY Printer	HP 7906 Disk Driver, DVR 32
Photo Reader	RTE-III WHZAT Inquary Program
256 Word WCS	HP Assembler and XREF
	RTE FORTRAN IV Compiler
	RTE FORTRAN IV Formatter
	DOS/RTE Relocatable Library
	Multi Terminal Driver, DVR 05
	Line Printer Dirver, DVR 12
	Batch Monitor Program
	Batch Monitor Library
	Memory Resident Programs
	Disk Resident Programs
	Writable Control Store (WCS) Driver, DVR 36
	RTE Microassembler, % MICRO
	RTE Micro Coss Assembler, % MXREF
	RTE Micro Load Utility, % WLOAD

TABLE III

AFIT RTE-III System Assignments and Driver Layout Used

Device or Accessory	Driver is	Uses DMA	Buffered output ?	Timeout value ?	EQT Extension	I/O Slot ?	LU #	EQT #	Sub-channel	Comments
WCS (12978A)	DVR36	NO				10	15	3	2	256 word
Time Base Gen. (12539C)						11				
7[illegible]6 Disk	DVR32	YES				12	2 21 20	1 1 1	1 2 0	lower lower upper
System Console 2648	DVR05	NO		1000	X=13	13	1 4 5	2 2 2	0 1 2	console L CTU R CTU
Microcircuit #1	n/a					14				
Microcircuit #2	n/a					15				
A/[illegible] converter	n/a					16				
I/[illegible] Photoreader	DVR01	NO	YES	500		17	11	5	0	
Ti[illegible] TTY/Printer	DVR00	NO	YES	18000		20	10	4	0	

After generating the new system configuration an off-line system backup was performed using the HP utility program !DISKUP. It is a wise practice to back up the operating system especially when microprogramming development is being done. The reason is that a microprogramming mistake can be very costly. Microprograms have complete control over the computer, and it would be relatively easy to make a mistake in programming which could easily destroy the operating system or other files. Having a back up system on a spare disk under these circumstances is absolutely essential. A list of the commands used to run !DISKUP is included in Appendix F. More detailed information on !DISKUP is found in Ref 23.

Microprogramming Software Support

One of the most important features of the microprogramming capability offered by the HP system is the development support software available. This software package consists of six programs:

1.	A MICROASSEMBLER	&MICRO
2.	A CROSS REFERENCE GENERATOR	&MXREF
3.	MICRO DEBUG EDITOR	&MDEP
4.	LOADER PROGRAM	&WLOAD
5.	PROM PROGRAM TAPE GENERATOR	&PTGEN
6.	DRIVER PROGRAM	&DVR36

The WLOAD and DVR36 programs are required to be generated into the system. The other programs may be loaded at any time. For the system implemented at AFIT the MICRO, MXREF, WLOAD and DVR36 were generated into the system. The MDEP program was excluded from the

initial system generation since the on-line generator program halted several times at this program. The MDEP program was loaded into the system after system generation.

Detailed information concerning the microprogramming support software is contained in the following references (Ref 21, Ref 24, and Ref 25).

Problems Encountered

The major problem encountered during the implementation and system generation phase was the intermittent and random halting of the computer whenever certain system programs were run. The halting would occur while trying to run the FORTRAN compiler, assembler and especially the On-Line generator program (RT3GN). The reason for why the halting occured was not discovered. Diagnostic tests were run on the computer memory. No problems were found. The memory appeared to be working properly. However, it was found that after cleaning the memory board contacts the halting was significantly reduced. And after having generated a new system configuration for microprogramming the halting was nearly eliminated except when the on-line generator program was run.

It was learned that other HP21MX computer users on-base also have had similar halting problems. Their solution has been to have HP service the computer and install new memory boards. The underlying cause of the halting however has not been completely resolved. The author personnally experienced this problem on the AFWAL 21MX computer running with the RTE-IVB operating system. The AFWAL computer halted while trying to use the FORTRAN compiler.

Appendix C discusses other problems encountered during the implementation phase.

Results

The results of this phase were positive with respect to implementing a working RTE-III system and a working microprogramming capability. The WCS was installed and checked out. The support software was exercised and found to work fine. In fact the MICROASSEMBLER and other micro support software seemed to work better than the FORTRAN compiler and the ASSEMBLER. No problems were encountered with using the MICRO, MXREF, MDEP, WLOAD, and DVR36 software. Likewise no halts occured when using these software features. The PROM tape generator software was not tested.

Summary

This section briefly discussed the 21MX computer system and the system generation required to implement the microprogramming capability on AFIT's computer. The WCS feature was discussed and the HP support software highlighted. The major problem encountered during this phase was the computer randomly halted whenever system programs were run. The overall result of this phase was the successful implementation of a user microprogramming capability on AFIT's HP 21MX computer.

V. A MICROPROGRAMMED EXAMPLE: FAST FOURIER TRANSFORM

Rauscher's Law: "Microprogramming an inefficient algorithm does not make it efficient" (Ref. 26)!

Introduction

In chapter II it was determined what to microprogram in an application program, is a function of many things and first requires a critical analysis of the programs activity. In this chapter, for purposes of demonstration and analysis, two similar radix 2 fast Fourier transformer (FFT) FORTRAN subroutine (Ref. 27, Ref. 28) are examined for program activity. A limited prototype microprogram for performing the bit reversal algorithm is developed to explore the feasibility of employing microcode to speed up the FFT program execution time. Results indicate that at least some aspects of the microprogramming process can be affected successfully, although futher research is needed to develop more comprehensive procedures and to address other technical issues.

Fast Fourier Transform Programs

Two fast Fourier transforms programs, named FOUR1 and FOURE, were chosen as suitable programs to demonstrate the concept of applying microprogramming to speed up execution response time. Both routines perform the well known Cooley-Turkey fast Fourier tran form (FFT) algorithm for a point sequence of complex numbers, x(m); m=o, 1..., N-1, where N must be a power of the, $N=2^m$. The FOUR1 subroutine was a program available on the EWAF system (Ref. 28). The FOURE subroutine was a program adapted from Radar's paper (Ref. 27). The FOURE subroutine

was chosen because it has a similar number of FORTRAN statements as the FOUR1 program but is coded differently. Radar recommends the FOURE program should be used only for demonstration purposes since faster routines are available. It was expected, then, the FOURE program would be less efficient than FOUR1. If so, it would be interesting to examine this difference when performing an activity profile analysis.

Method

To checkout and test the FFT programs it was convient to adapt Radar's main program FOURSUBT and modify it to include the FOUR1 subroutine.

Radar uses a well known sequence and deterministic closed form solution to test the FFT programs. The sequence he uses is:

$$X(n) = Q^n \qquad n = 0, 1, \ldots, N-1 \tag{1}$$

Where Q is a complex constant 0.9 + j0.3

and the closed form solution is:

$$X(k) = \frac{(1-Q^N)}{(1-QW^k)} \qquad k = 0, 1, \ldots, N-1 \tag{2}$$

$$\text{with } W = e^{\frac{-j2\pi}{N}} \tag{3}$$

This test permits a simple check of the FFT outputs with the data results published in Radar's paper for $N=2^5$ and also allows a comparison of accuracy obtained with the 21MX computer.

A simple timing subroutine was added to the main program to get the 21MX system time before and after each FFT subroutine was called in the main test program. This permited a simple way to determine how long it took each FFT subroutine to execute. The test program

was also made interactive with the HP graphics terminal so any number of points N=4, 8,..., 512 could be tested by just entering the value.

The programs, FOUR1 and FOURE, were first checked out on the EWAF RTE-IVB system and then later on the AFIT RTE-III system after it was made operational. Two observations were made by this baseline check of the FFT programs. First it was found the FOUR1 program is about twice as fast as the FOURE program. This faster response was expected as previously mentioned. The second observation was, however, unexpected. The accuracy was better on the EWAF 21MX (2112) computer using RTE-IVB system than on AFIT's 21MX (2108) computer using RTE-III system. Table IV shows the maximum absolute differences obtained between the theoretical closed form solution (Eq 2) and FOURE FFT values and the maximum differences obtained between original data sequence values (Eq 1) and the Inverse FOURE FFT data for the case of $N = 2^5$. Also shown for relative comparison is the results obtained by Radar on a Honeywell 6080N computer. The difference in accuracy between the EWAF 2112 and AFIT 2108 computers is attributed to several possible sources:

- Differences in the system operating software since the EWAF uses newer versions -- Possibly the SINE routines are more accurate.
- The instruction set in the EWAF is probably a newer version than in the AFIT system and may use more algorithms.
- Other differences in Operating system.
- Different FORTRAN compiler.

The actual cause of these differences was not pursued further. It is not the intent of this study nor within the scope to describe the theory or possible applications behind the FFT algorithm. This theory is adequately covered in many references. See for example (Ref. 29, Ref. 30, Ref. 31). Rather the goal is to demonstrate those sections in the FFT program software that could be microprogrammed to increase response time.

FFT Profile Analysis

After verifying that the FFT routines worked, the next step was to run the Activity Profile Generator (ACTV) program to obtain an activity profile of each FFT subroutine. The general purpose of an activity profile was covered in Chapter II. A brief discussion on how to run ACTV under RTE-IVB and a copy of the FORTRAN source listing for the ACTV program is included in Appendix H.

Samples of the profiles obtained for the FFT routines both with and without the microcoded bit reversal routine are included in Appendix I.

An analysis of the FFT profiles was made by examining the activity addresses with the program load map and copy of the FORTRAN mixed code listing. This led to the general findings shown in Figure 2.

It was determined that the Bit Reversal routine differed significantly between FOURI and FOURE in terms of activity. The FOURI program averaged about 3.6% total activity while the FOURE program averaged about 17% of the total activity. The activity for the butterfly computations in each program were roughly the same, however, except for one observation. On close examination of the mixed FORTRAN code listing for

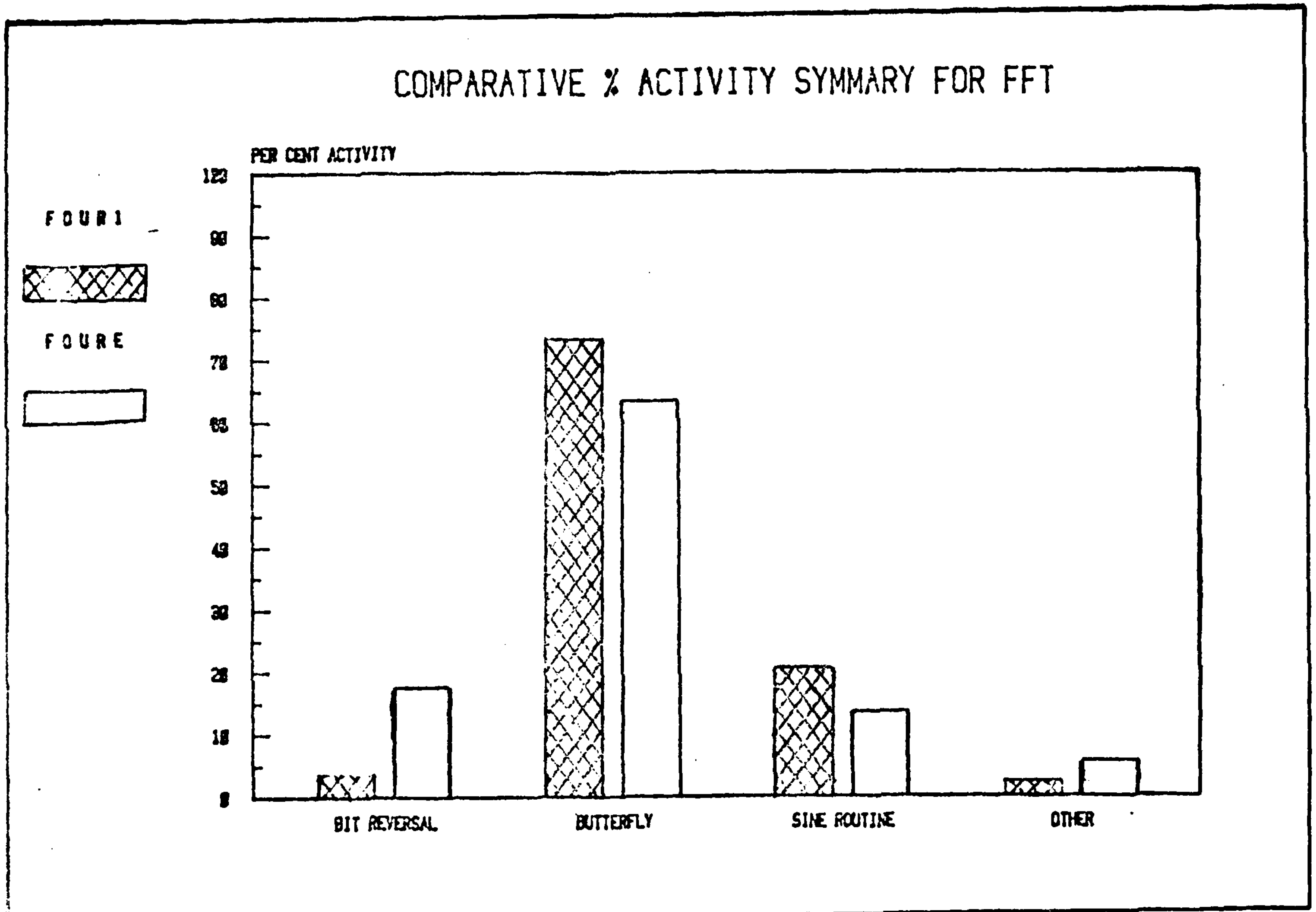


Figure 2 Activity Profile Summary for FOUR1 and FOURE FFT programs

TABLE IV

COMPARISON OF MAXIMUM DIFFERENCES

FOR FOURE FFT SUBROUTINE

($N = 2^5$)

Computer System	Max. Diff. Between Theoretical and FOURE FFT	Max. Diff. Between Original Sequence and Inverse FOURE FFT
Honeywell 6080N	0.238 10^{-6}	0.263 10^{-7}
EWAF HP 2112 RTE-IVB	0.534 10^{-6}	0.533 10^{-6}
AFIT HP 2108 RTE-III	0.111 10^{-4}	0.105 10^{-5}

the FOURE program, one arrives at the conclusion that approximately 18% of the total activity was spent in the execution of the two CONTINUE statements of label 110 and 120 in the program. The continue statements form the butterfly loops in the FOURE program. From the activity profile these two statements had 98 and 13 interrupts respectively out of a total of 584 hits for the entire subroutine. Hence approximately 18% of the FFT is spent here. The FOUR1 software structure is different from the FOURE program and does not exhibit this effect. The observation to be made here is that microcoding this short section of code in the FOURE program could speed up the FFT as much as microcoding the entire bit reversal algorithm which takes approximately 17% of the FFT time.

From Figure 2 it is seen about 20% of the execution time for the FOUR1 program is spent computing the SINE vlaues. Likewise about 13.5% of the activity in the FOURE program is spent computing the SINE values. The remaining activity shown in Figure 2 is spent in other parts of the FFT software code. This overhead software was not looked at in further detail for microcoding.

Predicted Improvement via Microprogram Enhancement

The results of the profile analysis shows roughly the amount of speed up one might expect by microprogramming the various sections of the FFT programs. Thus for example one might infer that microprogramming the software that performs the bit reversal should speed up the FOUR1 program roughly by 3%-4% and the FOURE program about 18%.

Microprogram Requirements

Now consider the requirements for microprogramming the various sections. The bit reversal is basically a sorting routine and involves no

computations as such. Unfortunately the results of the FFT activity profile analysis indicated microprogramming the Bit Reversal software will improve the overall speed only 3 to 18%, nevertheless, it is an example of a prime candidate for microprogramming.

On the other hand consider the Butterfly and SINE software. This code is mostly composed of complex multiplications, additions, subtractions and the SINE calculations. The floating point multiply and add and subtract routines are already microprogrammed by HP and are included in the base instructions set. Nothing is gained by microprogramming these routines by the user. This further suggests that it is possibly a waste of time to try to microcode this section since it is already running nearly as fast as microcoding will allow. The only exception is in the FOURE program. Close examination of the mixed FORTRAN/ASSEMBLER listing showed that the CONTINUE statements that end the butterfly code use about 18% of the total time.

Microprogramming the SINE computations would speed up the process as noted before but it would also require a very long microprogram. Hence, it is possible that it would exceed the 256 word WCS area available for this study. This estimation was based on observing the length of the microprogrammed SINE computations in the base instruction set for the E and F series HP-21MX computers.

Another consideration is that the 21MX architecture for the M-series has only one SAVE register in the control section. The SAVE register is used for microsubroutine jumps. In the M-series only one jump is allowed in a microprogram: jumps cannot be nested. This limits how a user can use the base instruction set. The user cannot, for example, jump directly to the microprogrammed floating point routines in the

base set. Because the floating point routines employ jumps to other microprograms in the base set, user microprograms cannot access them directly. This situation is corrected in the HP-21MX E and F series computers by providing three SAVE registers. If an E and F series computer were available it would be reasonable that an entire complex FFT algorithm could be coded into a 256 WCS board using jumps to the microprogrammed floating point and SINE routines in the base instruction set.

Design of the Microcode

As a result of the FFT profile analysis, limited WCS area and for purposes of demonstration it was decided to attempt to microcode the bit reversal sorting algorithm. Figure 3 shows a flow chart of the overall algorithm that was microcoded. The computation of the FFT butterfly was not microcoded. This section describes the design process for microprogramming the bit reversal algorithm.

The design was divided into two distinct parts: the bit reversal algorithm and the complex floating point data exchange. A listing of the entire microprogram developed is given in Appendix J.

The algorithm for forming the reversed index number was adapted from a microprogram listing for an integer FFT routine written for the HP 2100 computer (Ref. 32). The HP 2100 microcode unfortunately is incompatible with the HP-21MX computers thus it required considerable effort to learn the 2100 microlanguage to first make sense out of the algorithm and adapt it to the 21MX microprogramming language.

The complex data exchange section was first microprogrammed. The exchange section takes the complex floating point numbers stored in the

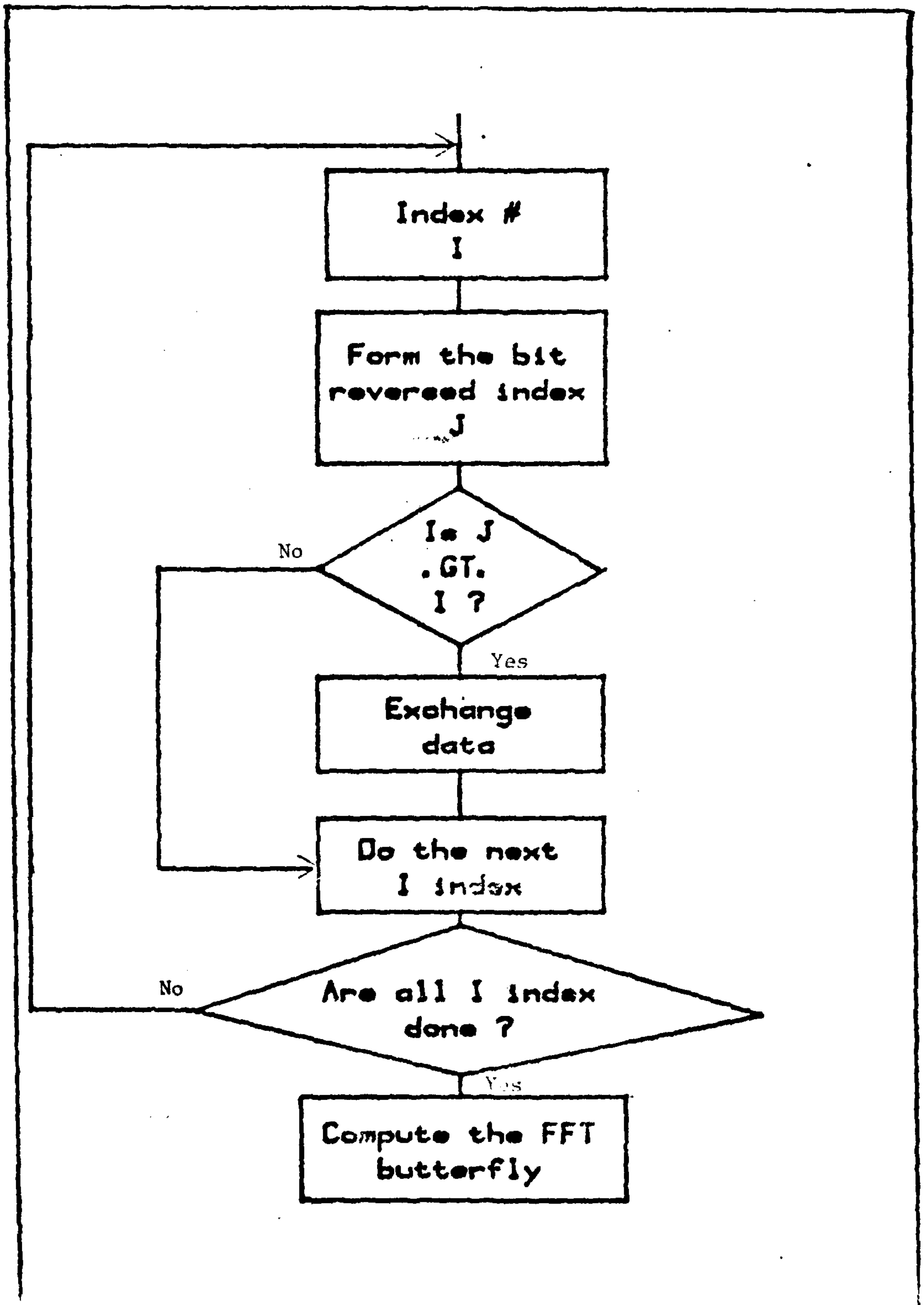


Figure 3 Flow Chart of Bit Reversal Sorting routine

data array and switches the values with the bit reversed index. After writing the exchange microcode section it was assembled and debuged using the HP microassembler and RTE micro debug editor (MDEP) software utility. The microcode for the bit reversal was written next and debuged. After each section was working properly they were combined into a single microprogram and stored as a disk file. The final step after writing and debuging the microprogram using MDEP was to write the necessary assembly language program to pass the required parameters and addresses from the FFT program and to initiate the microprogram's execution. The assembly interface routine is listed in Appendix J.

Bit Reversal Algorithm

The simplest way to bit reverse a number is simply to perform a right shift on the number and load it bit by bit into a second register.

1st. register	$\boxed{011010000}$	shift # right bit by bit
2nd. register	$\boxed{000010110}$	into a second register

Unfortunately, it is not this simple to do with the 21MX architecture. Ideally it should only take N shifts to accomplish the bit reversal for an N bit index number. But since it is not possible to do this directly with the 21MX microlanguage an algorithm must be used to perform the bit reversal.

Fortunately, the algorithm is simple and easily programmed in microcode using the extra storage registers available to the microprogrammer. The algorithm is as follows:

Step 1. Put the number to be bit reversed in a scratch register.

Step 2. Then examine the least significant bit (LSB) of the number.

Step 3. If the LSB is set then shift a bit into a second scratch register.

Step 4. If the LSB is zero then shift a zero into the second scratch register.

Step 5. Now repeat steps 2 through 4 and continue in this manner until all N bits of the number have been examined.

Step 6. The bit reversed number will then be in the second scratch register.

Microprogramming the Bit Reversal

The microprogram code to perform the above algorithm is illustrated in Figure 4. The microprogram takes 8 microinstructions. The microinstructions are in a loop, however, which repeats the microcode according to the number of bits in the number being reversed. The microprogram shown assumes the number to be reversed is in register S4. It assumes S2 is initially set to zero and will contain the bit reversed number when done. It assumes S5 initially contains the number of bits for the number being reversed.

Testing the Microprogram

After writing the microprogram it was microassembled using the MICRO software assembler and then loaded into the WCS using the MDEP program. The MDEP program was used to change parameters, microinstructions, register values, and run the microprogram from the system console to test and debug the program.

The major problem experienced with testing and debuging the program [illegible] program was

LABEL	OP-CODE	SPECIAL	ALU	STORE	S-BUS	COMMENTS
LOOP		L1		S2	S2	LEFT SHIFT S2 1 bit
				S4	S4	CHECK INDEX
	JMP	CNDX	ALO	RJS	*+2	ALO BIT-- JMP NEXT INST
*						IF ZERO
			INC	S2	S2	ADJUST BINVERT COUNT
		R1		S4	S4	RIGHT SHIFT INDEX
			DEC	S5	S5	ADJUST BIT COUNT NN=NN-1
	JMP	CNDX	TBZ		*+2	IS COUNT ZERO? YES,JMP
*						NEXT INSTRUCTION
	JMP				LOOP	NO!!! NOT DONE YET!!!

Figure 4 Microprogram to Form Bit Reversed Number

run under MDEP if a loop or faulty code occured or other error was made the computer would hang up in the microcode. Microprograms run all the time in the HP machine, even after the machine is halted. It was impossible to get out of a faulty loop in the microcode without turning the power off. When power is turned off of course, the WCS loses the microprogram and it must be reloaded. Fortunately it is simple to reload the code, provided it was first saved on a disk file. However corrections which were made to the program while debugging are lost and will need to be restored.

Microprograms can be loaded into WCS in several ways. Two ways were tried during this study. First the MDEP program can load the microcode stored on a disk file into the WCS. This method was primarily used during the debug verification phases of the program development. Any microcode loaded using the MDEP is static. This means the microprogram can not be changed as a software program is executing. It is equivalent to microcode residing in a PROM. The second method to load microcode into WCS is to use the WLOAD software utility program feature. This utility permits the WCS to be loaded from a calling FORTRAN or ASSEMBLY program. In addition it permits the calling program to dynamically load the WCS while it is executing. The dynamic capabilities of using this utility program was checked out by modifying the test program to interactively load the microprogram into WCS using the WLOAD feature. Using the WLOAD feature permitted loading the WCS interactively from the system console by simply specifying the file name containing the microprogram whenever the test program was run. Once the microprogram was loaded into WCS using this technique it was not changed during the execution of the test program.

Effectiveness of the Microprogrammed Bit Reversal

The effectiveness of the microprogram bit reversal sorting algorithm developed in this study was verified on both the AFIT 21MX 2108 computer and AFWAL's 21MX 2112 computer system.

Two measures of effectiveness were obtained. First, the average run time of the FFT programs with and without the microprogram was measured and the relative percent improvement in speed up computed. Second, a speed up factor for the microprogram itself was determined by comparing the measured average run time of the FORTRAN bit reversal software with the theoretical computed run time for the microprogram. The theoretical computation of the run time for the microprogram is given in Appendix K.

Figure 5 and Figure 6 show the comparative average baseline run time obtained for the FOUR1 and FOURE programs without using the microprogram. This data was taken on the AFIT, EWAF and AFWAL HP 21MX computers.

Figure 7 and Figure 8 show the comparative average run times obtained for the FOUR1 and FOURE programs when the microprogram is called. Data was obtained only for the AFIT and AFWAL computer. The EWAF computer does not have a microprogramming capability configured into the system. The AFWAL computer however had the microprogrammable writable control store module installed and the microprogram support software configured into their RTE-IVB operating system. It was therefore possible to run the same microprogram on the AFWAL computer to check the results obtained on the AFIT computer and also verify that microprograms developed at AFIT can successfully run on another HP 21MX computer system on base.

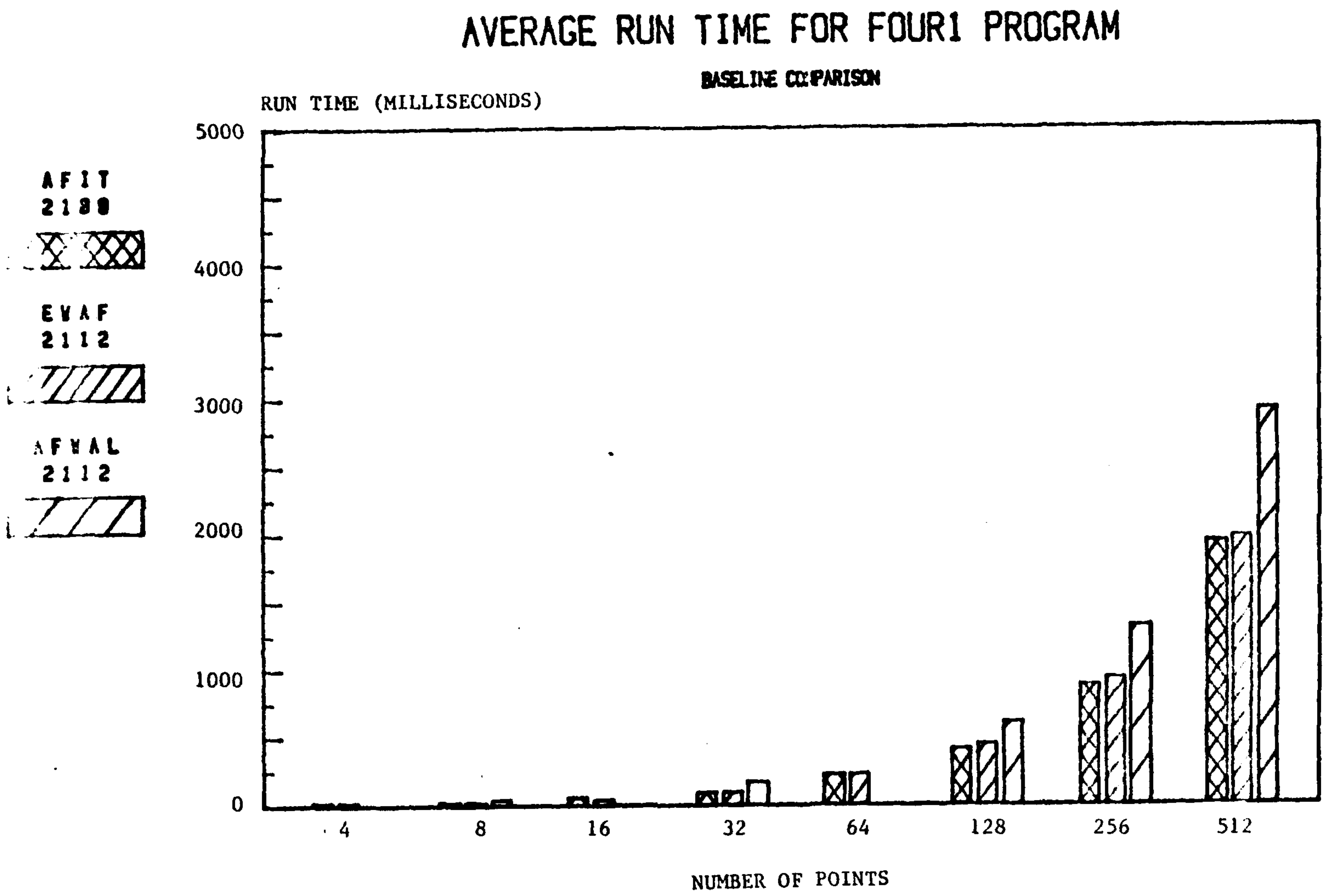


Figure 5 Comparison of Run Time for FOUR1 Program

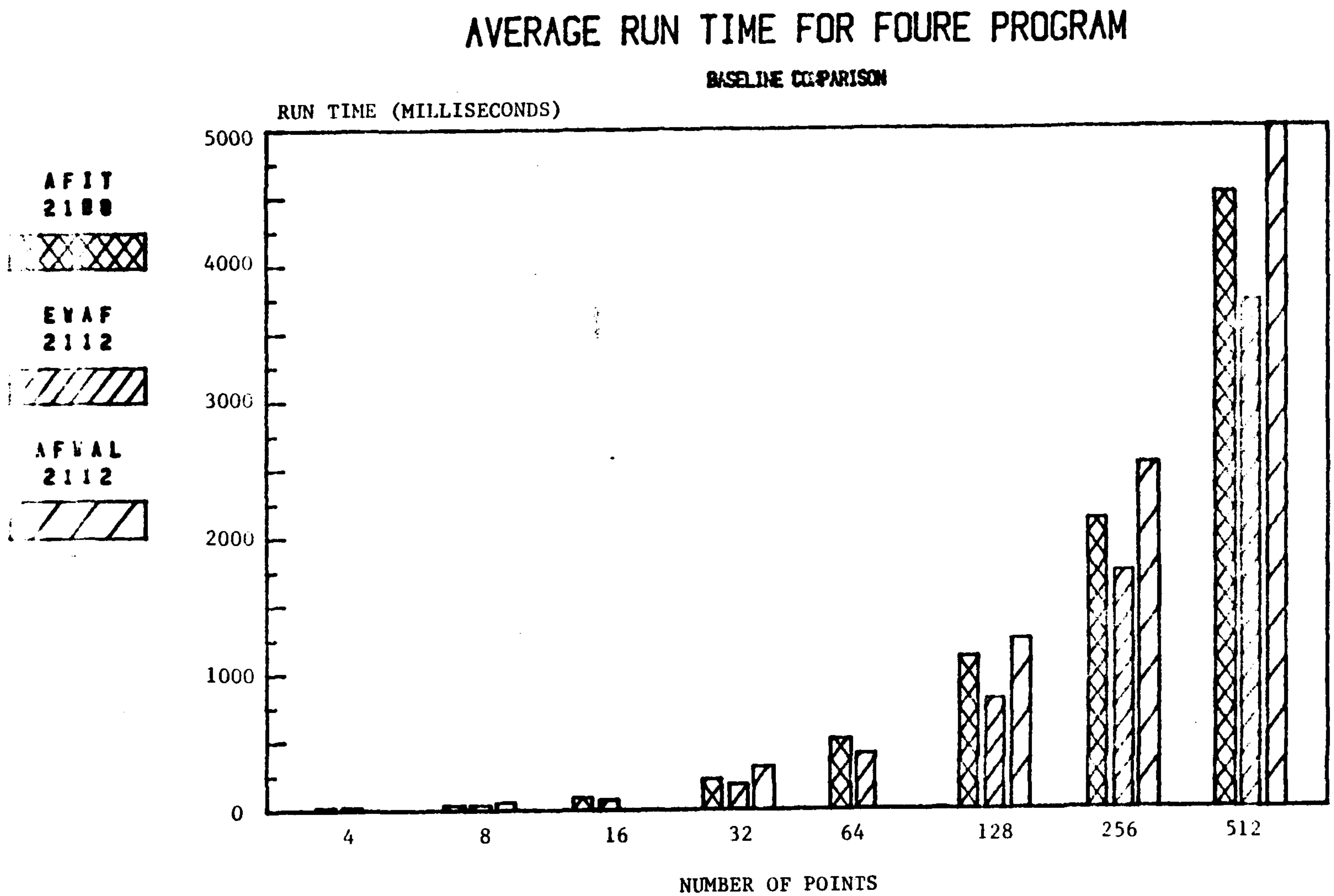


Figure 6 Comparison of Run Time for FOURE Program

TABLE V

Relative % Improvement Factor over Software FFT routines

Number of Points	AFIT 2108 Computer		AFWAL 2112 Computer	
	FOUR1	FOURE	FOUR1	FOURE
4	38	37	-	-
8	-6.97	16	-	-
16	13.3	6.4	-	-
32	4.45	2.44	-	-
64	8.99	1.13	-	-
128	-8.92	11.98	5.86	2.49
256	5.11	3.74	5.73	7.44
512	5.15	4.85	5.58	6.48

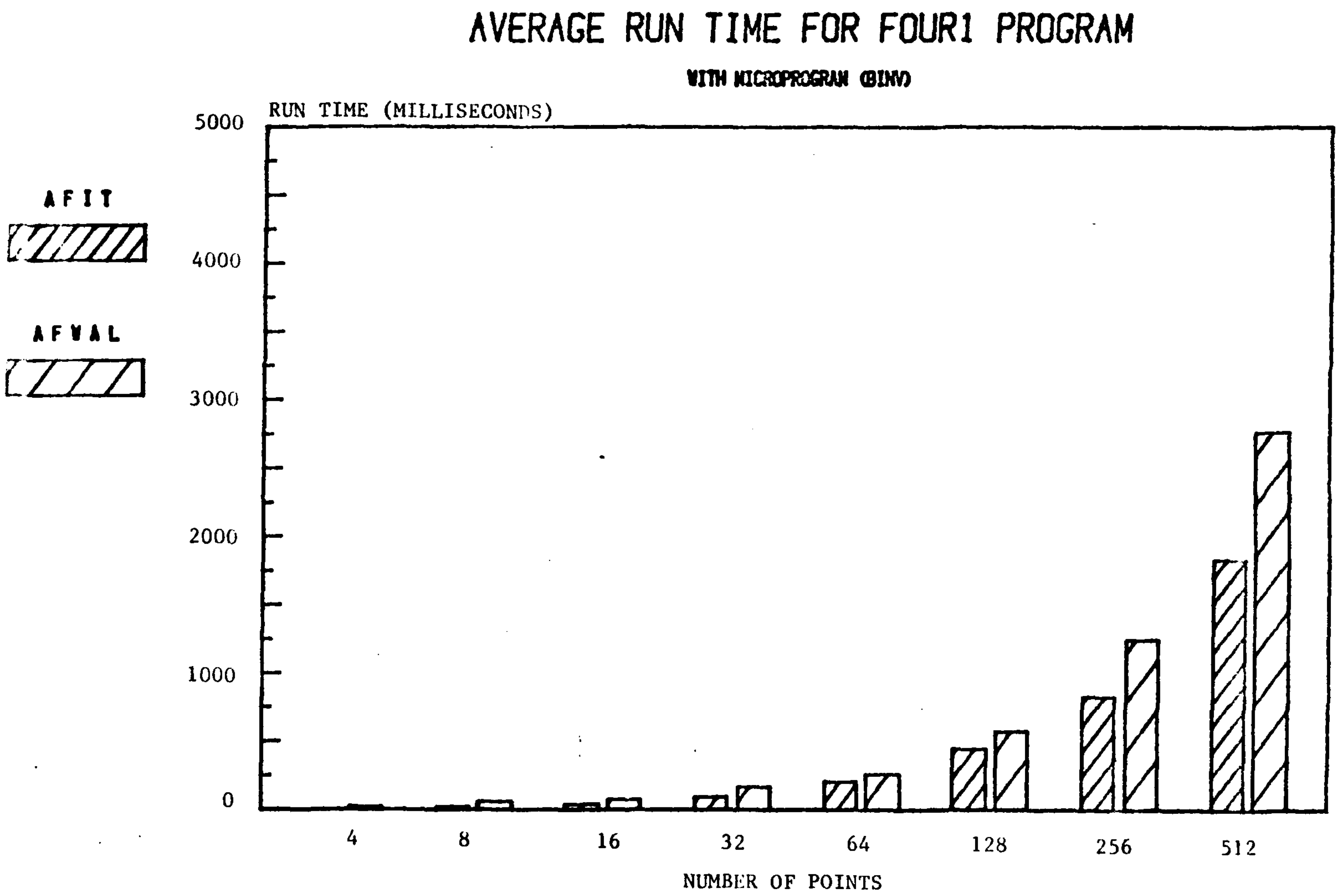


Figure 7 Comparison of Run Time for FOUR1 Program with Microprogramed Bit Reversal Routine

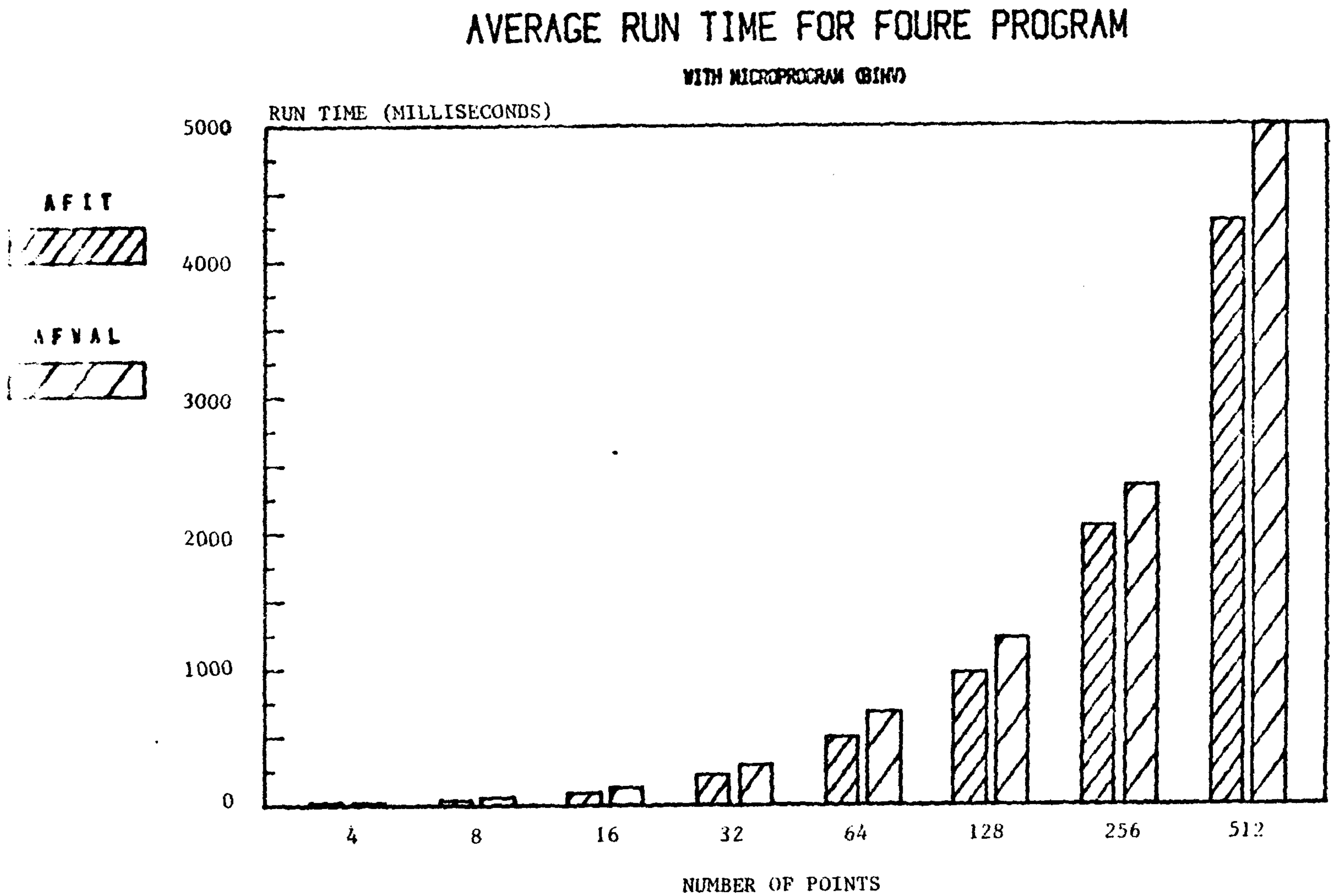


Figure 8 Comparison of Run Time for FOURE Program with Microprogramed Bit Reversal Routine

Table V shows the relative percent improvement in average run time obtained for the various number of points run for the FOURI and FOURE programs when the microprogrammed bit reversal routine is used. The table shows the improvement factor varies considerably for the different number of points tested. This is attributed in part to the imprecision in the method used to time the routines particularly for the cases where the number of points is less than 32 points.

The method of timing was accurate only to about 10 milliseconds by using the computers system clock. No conclusions are drawn on the timing results obtained for the cases having less than 32 points.

One discrepancy is apparent in table V however, for the 128 point case. In the 128 point case the timing should be relatively stable but the results showed a speed up did not occur on the AFIT computer system. This is contrary to what was expected. To check this discrepancy the same program was run on the AFWAL computer system. The timing results obtained indicated a 5.86% improvement for the 128 point case. This improvement was in line with what was expected from the profile analysis of the FFT. Likewise the factors obtained for the 256 and 512 point cases tend to stablize around 5-6 percent improvement which is roughly that expected from the activity analysis.

An attempt was also made to measure the average run time of just the FORTRAN bit reversal code. Figure 9 shows the results measured and compares them with the theoretical calculated run time for the microcoded bit reversal routine. Figure 10 shows the speed up factor of the microprogram over the equivalent software code. From Figure 10 it is clear that the microprogram is roughly 9 to 10 times faster than the software routine.

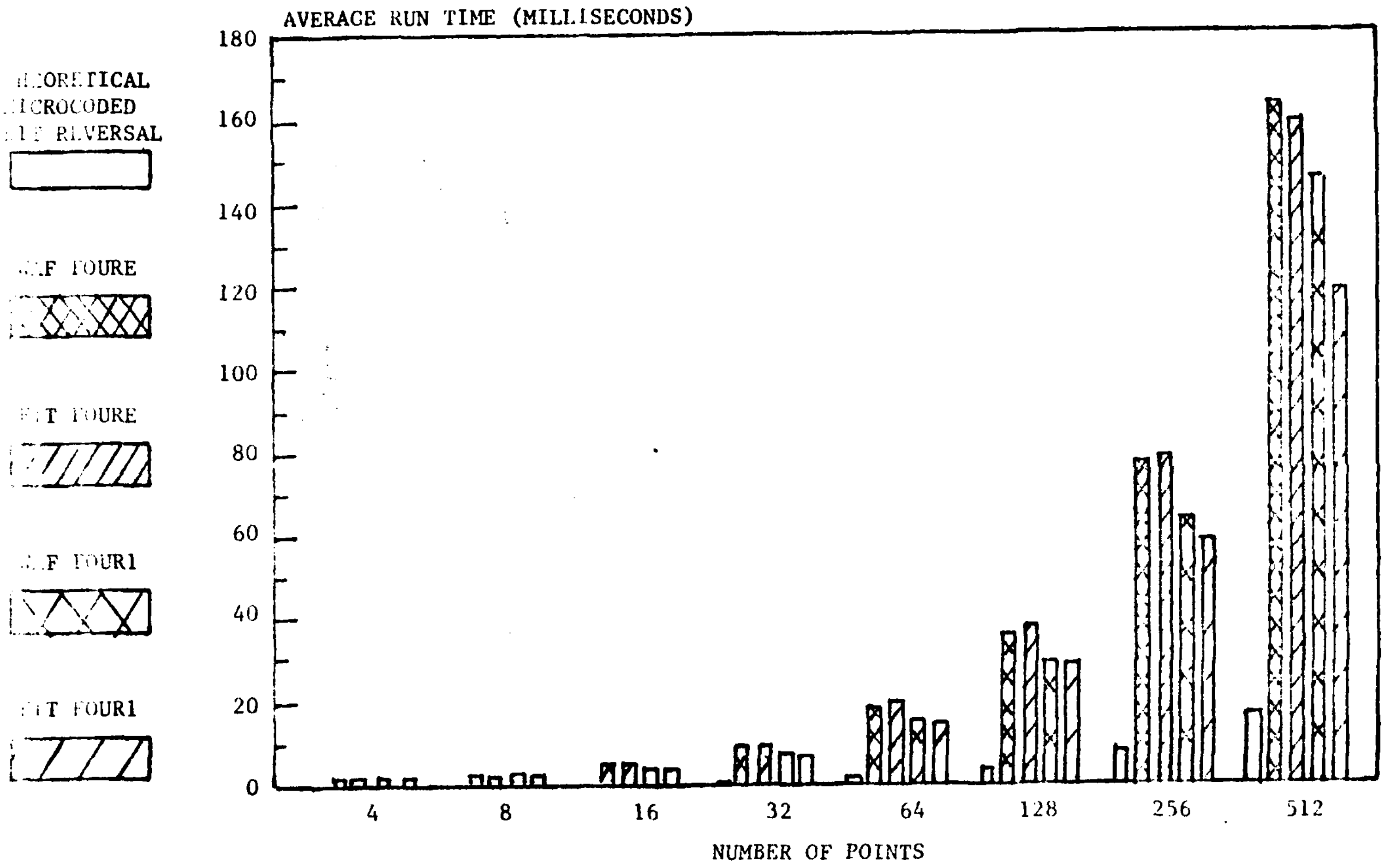


Figure 9. Comparison of Run Time for Bit Reversal Code

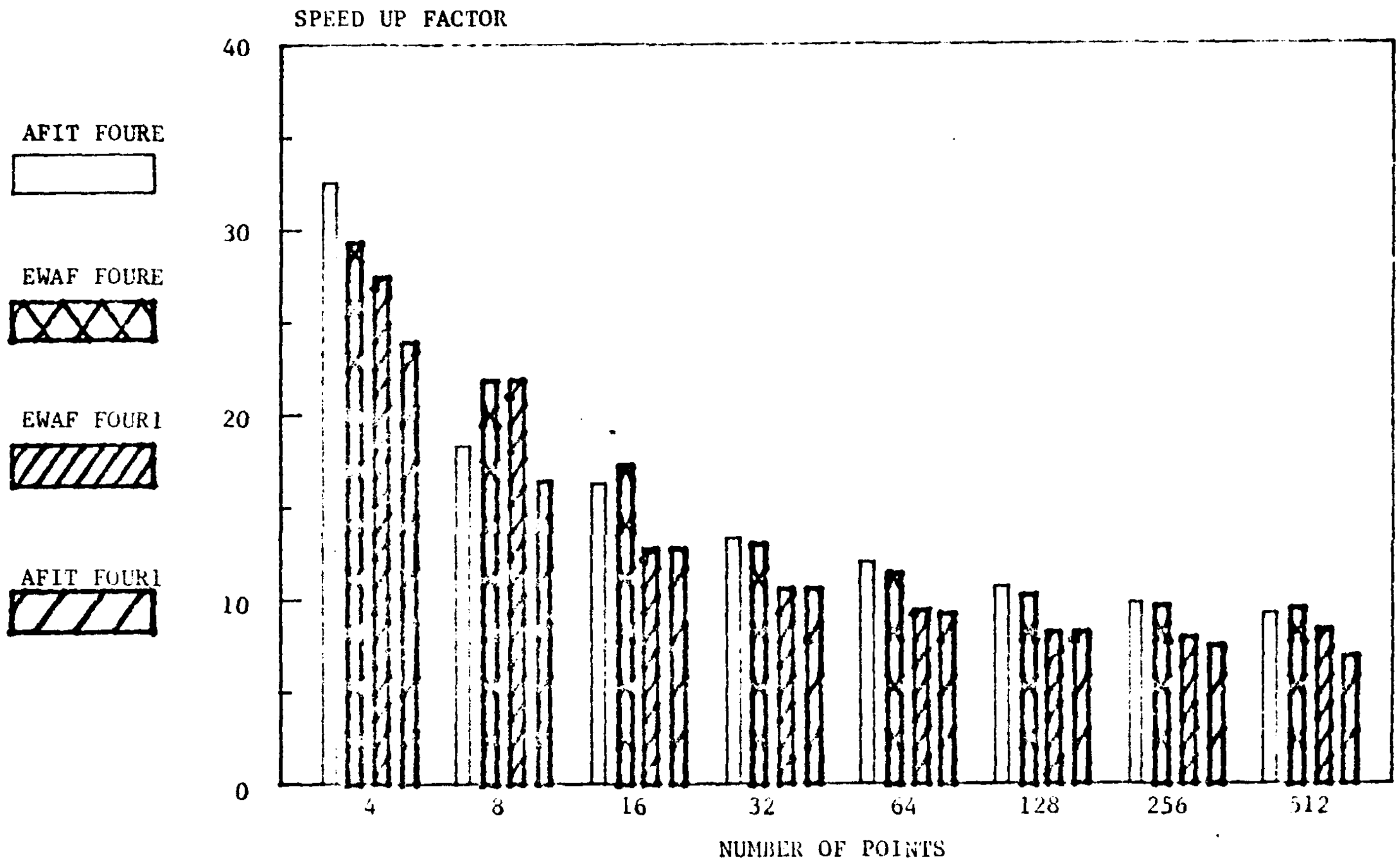


Figure 10. Comparison of Speed Up Factor for Microprogram

Summary

In summary, this section analyzes two similar fast Fourier transform programs FOUR1 and FOURE with the activity profile generator (ACTV) program. Analysis of the resulting activity profiles reveal microprogramming the bit reversal sorting section would improve overall performance approximately 4% to 17%. A limited prototype microprogram was developed for performing the bit reversal sorting algorithm. The resulting performance using this microprogramming was verified experimentally. The results show that the microprogramming did speed up the bit reversal algorithm itself by almost a factor of 10 but improved the overall response of the FFT programs only by about 5 - 6 percent.

VI. RESULTS AND CONCLUSIONS

Introduction

This section summarizes the results and conclusions arrived at during this study. This study is divided into three parts. First, the survey of electronic warfare programs on the ASD Electronic Warfare Analysis Facility (EWAF) and the selection criteria and analysis used to select a program for possible improvement using microprogramming techniques. Second, the implementation phase where the AFIT 21MX computer was assembled and a RTE-III system generation was done to implement a user microprogramming capability on the system. And third an actual demonstration involving the development of a microprogrammed bit reversal sorting algorithm for an FFT program and resulting tests to measure the relative speed gain achieved when the FFT program is run.

Results

The EW program survey resulted in a list of programs used for electronic warfare analysis and data reduction tasks. Most of the programs are currently running on the EWAF 21MX facility. Ssome of the larger simulation models however run on the base CDC computer and are being considered for implementation on the EWAF computer in the future. Likewise a number of digital signal processing programs (such as the fast Fourier transform programs) are of general interest to the EW community and would be useful to implement in real time. As a result of this survey several programs were examined for possible improvement in their execution speed. An activity profile of some of the programs showed that it would be difficult to improve certain programs which

exhibit high activity in the system library routines. As a result of this analysis it was decided to perform a more detailed profile analysis on the FFT programs available and to demonstrate the microprogramming concept by microcoding portions of the FFT programs.

It was found from the detailed profile analysis of the FFT programs that the bit reversal sorting took approximately 3% to 17% of the FFT's time. By microprogramming the bit reversal portion of the FFT it was demonstrated a speed up of approximately 5-6% resulted for the entire FFT routine. This is roughly that predicted by the profile analysis of the FFT. Further, measurements of the run time for bit reversal FORTRAN software compared with the theoretical microprogrammed bit reversal run time resulted in a speed up of 9 to 10 times over the same function coded in FORTRAN.

The bit reversal algorithm microprogrammed in this case demonstrates several important points: one, it shows that microcoding does decrease execution time. Two, it shows also that what is microprogrammed does not necessarily improve the overall response of a program. Three, the overall improvements to a program can be reasonably predicted through analysis using an activity profile generator. Four, it illustrates the complexity and attention to detail the user must deal with when using microprogramming. Five, it illustrates the limitations imposed by HP-21MX M-series computer when considering the use of the floating point routines in user microprograms. Six, it shows the flexibility offered by HP microprogramming.

Conclusions

1. The study objective of implementing a dynamic user microprogramming capability at AFIT was accomplished. The AFIT computer is fully capable of supporting user microprogramming research as was demon-

strated by developing and executing a microprogrammed bit reversal algorithm for an FFT program.

2. An activity profile scheme has been shown to highlight a programs performance from which a user can easily deduce the significance microprogramming will have on performance.

3. Microprogramming is specialized and more difficult than assembly language programming and is worth learning but its benefits are higly dependent upon the programs application. It can provide a powerful tool for solving time critical applications but at the same time it is not a good tool for everyday general applications.

4. It was demonstrated that AFIT has the capability to support user microprogramming for the HP-21MX computer systems and could provide a significant focal point for HP users at Wright-Patterson through AFIT research on project involving microprogramming.

Recommendations

Based on the results of this study the following recommendations for further study are submitted:

1. Continue investigating the capabilities user microprogramming offers especially for increasing throughput in a multi-user environment. One area observed in this study which could greatly effect throughput in a multi-user environment was microprogramming the HP RTE system library and utility programs. Another area for investigation could be the possibility of a distributed system. AFIT currently has two 21MX computers and is considering buying the F-series 21MX computer. One computer could be used to monitor the multi-user operating environment and it could pass off time critical programs to a second computer for faster

real time response. Microprogramming could be used here not only to speed response but also to taylor the interface hardware for maximum utilization.

2. Expand the basic understanding and procedures outlined herein and investigate desirable algorithm sets which will optimize the execution of programmed applications. For example various algorithms which perform the FFT using integer operations are faster than floating point routines. Investigate the drawbacks and possible advantages of user dynamic microprogramming and expand this to include the effects of multi-users on the system. A related area of study is the use of higher order language compilers to directly translate programs into microcode. A final area of study includes development of better software support tools. HP software support is excellent for microprogram development but it could be improved. For long microprograms it would be useful to develop an activity profiler to monitor microprograms. This would permit studying the activity within microprograms being developed.

3. Recommend that AFIT upgrade the existing 21MX M-series computer to an F-series computer. In addition, consideration to obtaining a 9-track tape unit, 4 pen plotter and graphics printer should be pursued. The F-series computer is a much more powerful computer and has the floating point routines implemented in hardware. In addition the F-series can use more WCS than the M-series and is therefore much more flexible and tailorable. The microprogramming capability has been improved over the M-series and allows for multiple microsubroutine jumps which limited the M-series. A magnetic tape unit is necessary to interface with various labs and the CDC computer system on

base. For example, data acquisition results stored on tapes from other facilities can easily be processed. The plotter and graphics printer are required to put computed results into useful form for analysis and for real time monitoring tasks.

Bibliography

1. Agrawala, A. K. "Microprogramming: Perspective and Status," IEEE Catalog No. 76 CH1158-5: 3-3, 3-23.

2. Microprogramming: A Way to Get Higher Performance from HP-1000 Computers. Product Application Note 281-1. Hewlett-Packard Corporation. Sept 1978.

3. Cline, Ben E. "An Introduction to Microprogramming," BYTE: 210-217 (April 1979).

4. Wilkes, M. V. "The Best Way To Design An Automatic Calculating Machine," in Machester Univ. Comput. Inaugan. Conf.: 16, 1951.

5. Oberzeir, Joseph A. "Writable Control Store Saves Microprogramming Time and Expanse," Electronics 52: 121-5 (June 21, 1979).

6. Vahlstrom, R. and M. Malone. "Evolution of Microprogrammed Input/Output Processing in One Processor Family," Computer Design, 15: 98-100 (January 1976).

7. Fung, Fergus K. and Willis K. King. "The Implementation of a User-Extensible System on a Dynamically Microprogrammable Computer," MICRO 10 PROCEEDINGS: 33-36 (October 5-7 1977).

8. HP 1000 E-Series and F-Series Computer Microprogramming. Reference Manual 02109-90004. Hewlett-Packard Co., July 1978.

9. Gordon, P. and J. Jacobs. "Microprogrammable Central Processor Adapts Easily to Special User Needs," Hewlett-Packard Journal: 7-14 (Oct 1974).

10. Coury, Fred F. "Microprogramming and Writable Control Store," Hewlett-Packard Journal 23 (11): 16-20 (July 1972).

11. Drake, Harris D. "Development and Application of Microprograms in a Real-Time Environment," Hewlett-Packard Journal: 15-17 (1977).

12. Snyder, David C. "Computer Performance Improvements by Measurement and Microprogramming," Hewlett-Packard Journal: 17-24 (Feb 1975).

13. Agrawala, Askok K. and T. G. Rauscher. Foundations of Microprogramming Architecture, Software and Application. Academic Press Inc. 1976.

14. Snook, Ted. "System Programming Tool Box for Microcomputers," IEEE COMPCON: 83-87 (Spring 1979).

15. Guha, R. K. "Dynamic Microprogramming in a Time Sharing Environment," MICRO 10 PROCEEDINGS: 55-60 (Oct 5-7 1977).

16. RTE Activity Profile Generator. Program Name: RAPG, Hewlett-Packard Plus/1000 Library Contribution, Date Code: 2001.

17. Hewlett-Packard Library of Contributed Software Catalog. Product Number 92082A.

18. Leonard, Jim. AFWAL/AARF. Technical Discussions 23-25 Jul 81.

19. Douvile, Phil. ASD/ENADC. Technical Discussions 8 Sep 81.

20. 21MX Computer Series Installation and Service Manual, Manual Part No. 02108-90006. Hewlett-Packard Company.

21. 21 MX M-Series Computer. RTE Microprogramming Reference Manual. Manual Part No. 02108-90032. Hewlett-Packard Company. 1976.

22. RTE-II/RTE-III On-Line Generator. Reference Manual Part No. 92060-90020: Hewlett-Packard Company, July 1977.

23. HP 92068A. Utility Programs Reference Manual. Manual Part No. 92068-90010. Hewlett-Packard Company.

24. HP 1000 E-Series and F-Series Computer Microprogramming Reference Manual. Manual Part No. 02109-90004. Hewlett-Packard Company: July 1978.

25. 21MX Micrprogramming Under RTE. Student Handbook. Part No. 22999-90132. Hewlett-Packard Company: March 1977.

26. Rauscher, Tomlinson, G. "Microprogramming: A Tutorial and Survey of Recent Developments," IEEE Trans. on Computers, c-29 No. 1: 2-19 (January 1980).

27. Radar, C. M. "FOUREA - A Short Demonstration Version of the FFT," in Programs For Digital Signal Processing, IEEE Press: 1.1-1, 1.1-5.

28. Kent, Bill. "The FAST FOURIER TRANSFORM: CALCULATIONS AND INTERPRETATIONS FOR SIGNAL APPLICATIONS," Technical Note ASD/ENADC, Wright-Patterson AFB OH. Aug 1976.

29. Cochran, W. T. et al. "What is the Fast Fourier Transform?," IEEE Trans. Audio Elecroacoust., AU-15: 45-55 (June 1967).

30. Openhein, Alan V. and R. Schafer. Digital Signal Processing. Prentice-Hall. 1975.

31. Brigham, Oran E. The Fast Fourier Transform. Prentice-Hall, Inc., 1974.

32. Koning, R. A. Micro Binversion Routine for a Microcoded FFT for the HP 2100 System (Source Listing Available at AFIT) Feb 1973.

APPENDIX A

List of Programs Surveyed

The following is a list of the post-data processing, digital signal processing, and electronic warfare analysis programs used by the Electronic Warfare Division, ASD/ENAD, Wright-Patterson AFB.

Description	Source
Data Transformation and Mathematical Programs	
1. Transformation Program (Like the BMD Transgeneration)	&TRANF
2. Matrix Arithmetic Program (Add, Subtract, Multiply up to 20 x 20 matrices)	&MRITH
3. Simultaneous Equation Solver (Up to 22 Equations, 22 unknowns)	&SIMEQ
General Statistics Programs	
4. General Statistics Information Program (Elementary calculation, point estimates, mean standard deviation, a histogram)	&STATS
5. General Statistics for Multiple Groups (Same as STATS but many groups at same time)	&MULTI
6. Paired T-Test Program (Student's-T for paired observations)	&PAIRT
7. Test of Hypothesis (Test Ho: $\mu = \mu_0$ or Ho: $\mu_1 = \mu_2$)	&TTEST
8. Moving Averages in A Time Series	&MOVAV
9. Cross Tabulation Program (Cross Tabulation on Two Single Dimensional Fixed Point Arrays)	&CRTAB
10. Discriminant Analysis Program	&DSCRA
11. Sample Size Determination on the Sample Variance (Estimates sample size from S^2 and degrees of freedom)	&SAMPL

Description	Source
12. Confidence Interval for Mean and Variance of a Normal Distribution	&CONFI
13. Cumulative Distribution Program	&CUMDS
14. Test of Hypothesis for Variance (Test Ho: $\sigma^2 = \sigma_0^2$ or Ho: $\sigma_1^2 = \sigma_2^2$)	&VHTOH

Specialized Statistic Programs

15. X-4 Plotter on Printer (Plots on a standard Printing device Not Plotter)	&XYPLO
16. Histogram Plotting Program (Plots on a standard Printer not a Plotter)	&HSTPL
17. Bartlett's Homegeniety of Variance Test	&BARTL
18. Duncans Multiple Range Test	&DUNCN
19. Wilcoxon-Mann-Whiting Test	&WILMA
20. Kendall's Coefficient of Concordnace (Checks for Ties)	&KENDW
21. Kendall's Coefficient of Concordance (No check for ties)	&KENDL
22. Kendall's Tau Correlation	&KETAU

Cross Correlation Programs

23. Cross Correlation Program	&CROSCO
24. Multiple Correlation Program (row Sums of Squares Cross Product Matrix, etc)	&CORRL
25. Multiple Correlation Matrix (Pearson Correlation Coefficient)	&CORMA
26. Auto Correlation and Spectral Analysis	&AUTSP

Regression Analysis Program

27. Least Squares Regression	&LSQRG
28. Linear Regression With Replication	&LINRE
29. Bioassay Program	&BIOAS

Description	Source
30. Linear Regression Confidence Interval Estimates (Least Squares)	&LSRCI
31. Polynomial Regression (Up to 15th degree Polynomial)	&LSRCI
32. Polynomial Regression with Confidence Intervals	&PLYCI
33. Orthogonal Polynomial Regression	&OPOLY
34. Non-Linear Regression Program (Least Squares Regression)	&NLSQR
35. Non-Linear Regression of an Arbitrary Function	&OMEGA
36. Non-Linear Regression of a Single Variable Function	&SOMEG
37. Stepwise Regression	&STEPR
38. Multiple Regression	&MULRE
39. Pooling of Groups in Regression	&POOL

Test of Fit Programs

Description	Source
40. Chi-Square Goodness of Fit Test (For normal Distribution)	&XIFIT
41. Kolmogorov-Smirnov Test of Fit	&KSTOF

The ANOVA Programs

Description	Source
42. Completely Randomized Design	&ANOV1
43. Completely Randomized Design with Subsampling	&ANOV2
44. Randomized Complete Block Diagram	&ANOV3
45. Randomized Complete Block Design with Subsampling	&ANOV4
46. Two Way Factorial Design	&ANOV5
47. Three Way Factorial Design	&ANOV6
48. Analysis of Variance Info Generator	&ANOV7

Subprograms in Stat Pack

Description	Source
49. Matrix Arithmetic Subroutine (Like [illegible] but s Subroutine)	&MARTH

Description	Source
50. Simpson Integration Subroutine	&INTGR
51. Matrix Inversion Subroutine Contains:	&MATIV
SYMIV - Symetric Matrix Inversion	
PINIV - Maximum Pivoltal Element Inversion	
QMINV - Short Cut Matrix Inversion	
SPMAT - Check for Significance of Pivotal Elements Before Inversion	
MATIV - Matrix Inversion with Simultaneous Equation Solver	
52. Histogram Plotting Subroutine (Like HTSPL but Subroutine)	&HPLSB
53. Simultaneous Equation Solver Subroutine (Like SIMEQ only Subroutine)	&SMSUB
54. Probability Subprograms Contains:	&PROSU
ANORM - Normal Distribution Function	
BINOM - Binomial Distribution Function	
POIS - Poisson Distribution Function	
FPROB - F Cummulative Probability Function	
CHIFN - Chi-Square Cummulative Probability Function	
CHISQ - Chi Square Subroutine	
FDIST - Inverse F Distirbution Function	
TDIST - Student's T Distribution Subroutine	
XAREA - Normal Probability Function	
55. Time Series Plotting Function (On Printer Not Plotter)	&SPLTR
56. Normal Distribution (Assembler Language Subroutine)	&PROBN

Description	Source
57. Variance Ratio Distribution Function	&PROBF

Electronic Warfare Analysis and Simulation Models

Description	Source
58. Programs used in Analysis of down link jamming	&DOALL
59. Model Simulates the intercept of one or more targets by an Air To Air, Surface to Air, Air to Surface or Surface to Surface Missile	GEMM
60. A missile, targe, and radar simulation -- Simulates Blast-Frag warheads.	BETA
61. Model for AWACS jamming radar and expendable jammers	EXPAND
62. Model calculates propagation loss and profile of transmitters and antenna in jungle conditions	COMTE
63. Models ideal three channel monopulse radar	SPACE
64. Models IR missiles and simulates ideal tracking	DECOYAN
65. Models IR missile simulation	HOME
66. Models IR missile simulation	ATEM
67. Models surface to air missile	TWS4
68. Models radar range and jammer equations for multi-jammer sources - computes jamming to signal (J/S) ratios	Expendable jammer model
69. Models effect of chaff	SCARE
70. Generates multi aircraft flight path scenarios	MPASS1
71. Terrain/Clutter model for MPASS3	MPASS2
72. Multi-element radar/target/jammer simulation program	MPASS3
73. A command and control campaign model	MECCA
74. Model to compute harmonics and intermodulation Products	Receiver model
75. Model for general theory of diffraction	GTD
76. Model computes bessel functions	&BESDV

Description	Source
77. NASA TWT program-simulates TWT response	TWT program
78. Various antenna data bank and data reduction programs to catalog, format and interpolate RADC antenna measurement data	
79. Marcum Swerling models	

APPENDIX B

A FORTRAN program named ACTV was used in this study to generate the activity profiles for analyzing where to apply microprogramming. A listing of the ACTV program is given below. The program was written to work with HP's RTE-IVB operating system. The ACTV program basically monitors the execution of an application program by periodically interrupting the application program and storing the contents of the program address register in 52 address cells. The rate of interruption is variable in integer increments of 10 milliseconds. The program allows the user to interactively specify the address range to be monitored and permits the user to zoom in on a specific address range of interest. The ACTV prints out a histogram and cummulative distribution curve of the activity monitored.

Appendix H contains a brief discussion on how to run ACTV.

```
FTN4,L,B
      PROGRAM ACTV
      DIMENSION FBF(52),IPR(52),IN(3)
C     ACTIVITY PROFILE GENERATOR USING SUSPEND ADDRESS
C     FROM ID-SEGMENT.
C        BY JIM LEONARD
C        USAF AVIONICS LAB, WPAFB
C
    5 WRITE(1,10)
   10 FORMAT(" ACTIVITY PROFILE GENERATOR ",//,
     1 " TYPE PROG NAME " )
      IN(1)=2H
      IN(2)=2H
      IN(3)=2H
      READ(1,20)IN
   20 FORMAT(3A2)
C     GET ADDRESS OF ID SEGMENT
      IDSEG=IDGET(IN)
      IF(IDSEG.NE.0)GOTO 100
      WRITE(1,30)IDSEG
   30 FORMAT("IMPROPER PROGRAM NAME, IDSEG= ",I8)
      GOTO 5
  100 WRITE(1,110)
  110 FORMAT("TYPE BOUNDS OF ACTIVITY PROFILE, LOWER-UPPER",/,
     1 " XXXXX XXXXX      X")
      NT=0
      READ(1,120)IL,IU,NT
  120 FORMAT(2K6,I6)
      IF(NT.LE.0)NT=3
C     INITIALIZE PROFILE BUFFER
      DO 130 I=1,52
      FBF(I)=0.
  130 CONTINUE
      ID=IU-IL+1
      INCR=(IU-IL+1)/50
      IF(INCR*50.LT.ID)INCR=INCR+1
      IW1=IDSEG+15
      IW2=IDSEG+[illegible]
C     IF PROGRAM IS NOT CURRENTLY ACTIVE DON'T RECORD LOCATION
  300 CALL RCORE(IW1,IVAL)
      IF(IAND(IVAL,15B).NE.1)GOTO 200
C     READ SUSPENDED LOCATION
      CALL RCORE(IW2,IVAL)
C     CKECK FOR BEFORE BOUNDS
      IF(IVAL [illegible]
      FBF(1)=FBF(1)+1.
      GOTO [illegible]
```

```
C      CHECK FOR BEYOND BOUNDS
 140   IF(IVAL.LE.IU)GOTO 150
       FBF(52)=FBF(52)+1.
       GOTO 200
C      MARK INTERVAL
  150  IVAL=(IVAL-IL)/INCR+2
       FBF(IVAL)=FBF(IVAL)+1.
C      TERMINATE MONITORING IF OPERATOR BREAKS
 200   IF(IFBRK(IDMY))500,210
  210  ISC=0
C      WAIT DESIRED INTERVAL
       CALL EXEC(12,ISC,1,0,-NT)
       GOTO 300
 500   WRITE(6,510)IN,IL,IU,INCR
 510   FORMAT("1PROGRAM ACTIVITY PROFILE FOR ",3A2,/,
      1 "    FROM",K8,"       TO",K8,"     IN INCREMENTS OF",I8)
 515   FORMAT(" INTERVAL NO.  FROM     TO     NO OF HITS    "
      1 ,"NORMALIZED HITS   NORMAL ACCUM")
C      FIND MAX VALUE OF HISTOGRAM
       FMX=-1.
       TSUM=0.
       DO 520 I=2,51
       TSUM=TSUM+FBF(I)
       IF(FMX.LT.FBF(I))FMX=FBF(I)
 520   CONTINUE
C      EXIT IF NO ACTIVITY IN DESIRED RANGE
       IF(FMX.GT.0.)GOTO 600
       IF((FBF(1)+FBF(52)).GT.0.)GOTO 540
       WRITE(1,530)
 530   FORMAT("NO PROGRAM ACTIVITY RECORDED--AT ALL!!!")
       WRITE(6,530)
       STOP
 540   WRITE(1,550)FBF(1),FBF(52)
 550   FORMAT(" NO PROGRAM ACTIVITY IN REGION OF INTEREST",/
      1   ,"BEFORE=",E13.7,"    AFTER="E13.7)
       WRITE(6,550)FBF(1),FBF(52)
C      WRITE TABLE OF ACTIVITY PROFILE
 600   WRITE(6,515)
       SUM=0.
       TSM1=TSUM+FBF(1)+FBF(52)
       DO 650 I=1,52
       SUM=SUM+FBF(I)/TSM1
       FNORM=FBF(I)/FMX
       IFR=IL+INCR*(I-2)
       ITO=IFR+INCR
       IF(I.EQ.1)IFR=0
       IF(I.EQ.52)ITO=32767
       WRITE(6,610)I,IFR,ITO,FBF(I),FNORM,SUM
```

```
 610  FORMAT(4X,I3,6X,2K7,F10.0,F17.8,F15.5)
 650  CONTINUE
C     PLOT HISTOGRAM ON PRINTER
      WRITE(6,510)IN,IL,IU,INCR
      WRITE(6,700)
 700  FORMAT(" INTERVAL  0          2          4          6          8"
     1  "          1")
C     FOR EACH DATA INTERVAL
      SUM=-FBF(1)/TSUM
      DO 800 J=1,52
C     CLEAR PRINTER BUFFER
      DO 710 I=1,51
      IPR(I)=2H
 710  CONTINUE
C     CALCULATE INDEXS
      SUM=SUM+FBF(J)/TSUM
      INDX=SUM*50.+1.5
      IF((J.NE.1).AND.(J.NE.52))IPR(INDX)=2HII
      INORM=50.*FBF(J)/FMX+1.5
C     PRINT AN X IF OFF PLOT
      IF(INORM.LT.1)INORM=-1
      IF(INORM.GT.51)INORM=-51
C     PRINT AN * IF ON THE PLOT
      IF(INORM.GT.0)IPR(INORM)=2H00
      IF(INORM.LT.0)IPR(-INORM)=2HXX
      WRITE(6,720)J,(IPR(K),K=1,51)
 720  FORMAT(2X,I6,3X,51A1)
 800  CONTINUE
      STOP
      END
      END$
```

```
ASMB L
      NAM RCORE
*     READS AND RETURNS THE CONTENTS OF A SINGLE
*     MEMORY LOCATION
*
*     THIS IS A SUBROUTINE TO THE ACTIVITY PROFILE GENERATOR
*     JIM LEONARD WROTE
*
*     THE ACTIVITY PROFILE SOURCE PROGRAM IS ON FILE &ACTV::34
*
*     JOHN STEIDLE
*
      ENT RCORE
      EXT .ENTR
IW1   NOP              ADDRESS OF ADDRESS OF DESIRED VALUE
IW2   NOP              ADDRESS FOR RETURNED CORE VALUE
RCORE NOP
      JSB .ENTR        GET PARAMETER ADDRESSES
      DEF IW1
      LDA IW1,I        READ ADDRESS
      LDA 0,I          READ CONTENTS
      STA IW2,I        STORE IT
      JMP RCORE,I
      END
```

APPENDIX C

The following is a list of on-base facilities using HP computer equipment and personnel surveyed for this project.

1.	Bill Griffin	ASD/ENAD	54061/52789	Bldg 125
2.	Jim Leonard	AFWAL/AARF-2	53050	Bldg 23
3.	Bob Ballard	AFWAL/FIMN	52593	Bldg 450
4.	Bryan Kent	AFWAL/AAWP	55076	Bldg 821
5.	Blenn Williams	AFWAL/FIMN	52493	Bldg 26/240
6.	Jim Mosora	AFWAL/MLPO	53808	Bldg 651
7.	Larry Linder	AFFDL	55205/56795	
8.	Dr. Gary Lamont	AFIT	53057	Bldg 620

APPENDIX D

The HP-21 M-Series User Microprogrammable Computer

Introduction

The purpose of this section is to review the structure of the HP 21MX M-Series Computer and to describe the HP software support tools available for microprogram development.

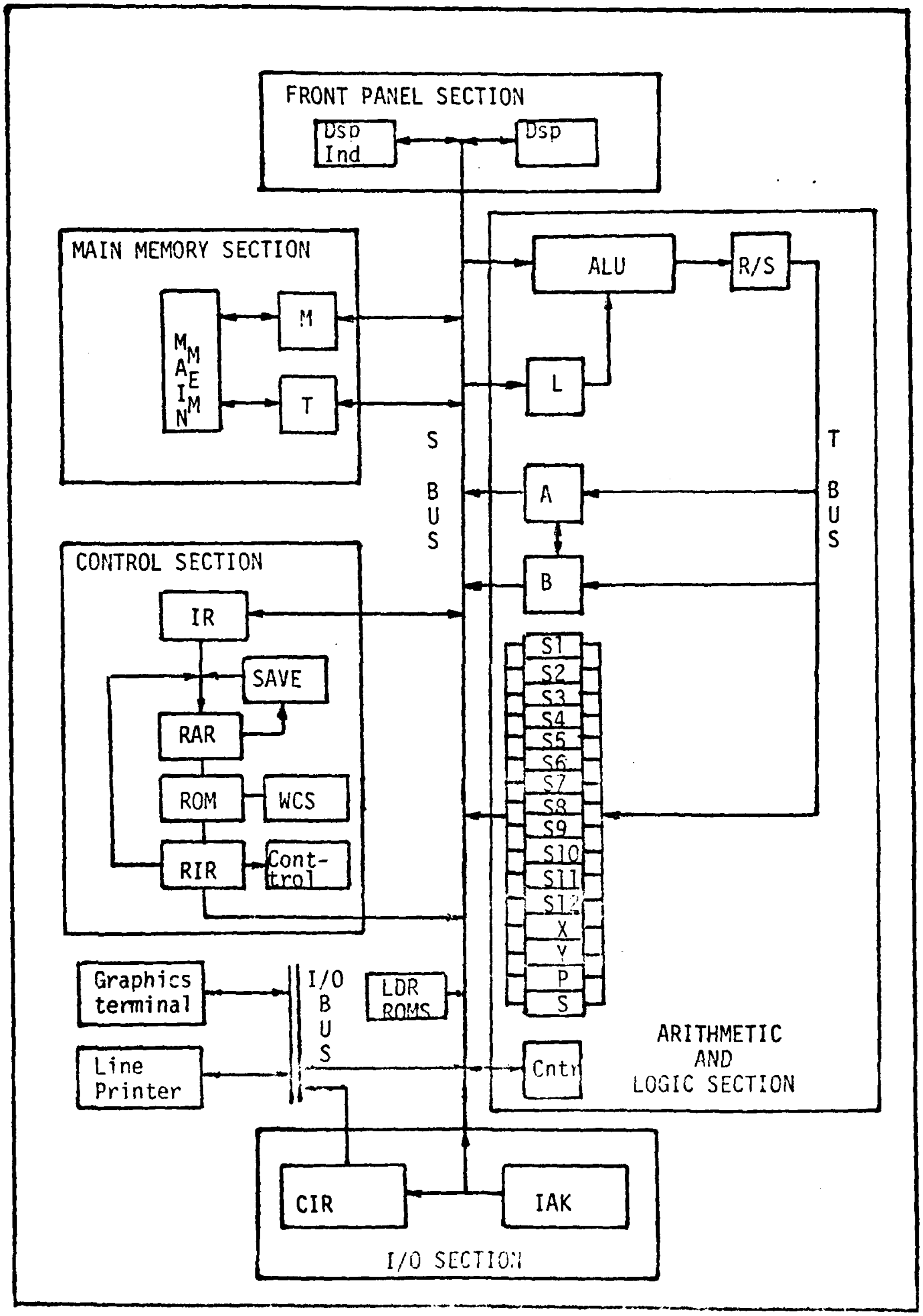
21MX Computer Description

The HP-21MX computer is a 16 bit, user microprogrammable general purpose minicomputer. Figure 11 shows a functional block diagram of the computers architecture. A good understanding of the architecture is a prerequisite to understanding and applying microprogramming effectively. The computer architecture is typical of most computers and can be broken down into the five general sections shown in Figure 11. The sections shown on Figure 11 are the arithmetic and logic section, computer control section, main memory section, front panel section and the input/output section.

Also shown on the diagram are the three main 16 bit data paths for the computer, namely the S-bus, T-bus and I/O bus. It can be seen that the various functional sections are connected via these buses. The S-bus is the central data transfer path. The S-bus can accept data from all registers except the L-register, SAVE register, RAR, Extend register and the Overflow registers. The Extend and Overflow registers are not shown on the diagram however. Any data placed on the S-bus is available to any of the following registers:

M-register

T-register


FRONT PANEL SECTION
Dsp Ind
Dsp
MAIN MEMORY SECTION
MAIN
MEM
M
T
S BUS
T BUS
ALU
R/S
L
A
B
CONTROL SECTION
IR
SAVE
RAR
ROM
WCS
RIR
Cont-trol
S1
S2
S3
S4
S5
S6
S7
S8
S9
S10
S11
S12
X
Y
P
S
Graphics terminal
Line Printer
I/O BUS
LDR ROMS
Cntr
ARITHMETIC AND LOGIC SECTION
CIR
IAK
I/O SECTION

L-register

CNTR (Counter register)

Display Register

Display Indicator

Instruction Register (IR)

The T-bus is a one way data bus. Its purpose is to receive data from the Rotate/Shifter (R/S) and selectively pass this data to any of the scratch registers (S1 through S12), the A or B registers, or the X, Y, P and S registers. The X and Y registers are often used as index registers. The P-register is used as the location address counter. The S-register is used for the front panel switch registers. The I/O bus permits transfering data to and from peripheral devices under programmed control. For example the graphics terminal or printer devices. The I/O bus interfaces directly with the S-bus.

Arithmetic and Logic Section

The arithmetic and logic section performs all arithmetic and functional modifications on the data for the computer. As pictured in Figure 11 this section is composed of three main units: the arithmetic and logic unit (ALU), the Rotate/Shifter (R/S) and 22 local registers. In addition, six flag registers are available (used to indicate special conditions) which are not shown on Figure 11. The ALU is designed to receive 16 bit data from the S-bus and from the L-register (latch register) and pass the resulting computation serially to the Rotate/Shifter (R/S) unit. A wide variety of shift operations can be performed on the data that leaves the ALU unit. The R/S puts the results onto the T-bus for assignment to one or more of the 22 local registers. The ALU operations include the following:

1.	ADD	6.	EXCLUSIVE OR
2.	SUBTRACT	7.	NAND
3.	MULTIPLY STEP	8.	NOR
4.	DIVIDE STEP	9.	INCREMENT BY ONE
5.	COMPLEMENT (NOT)	10.	DECREMENT BY ONE

Computer Control Section

The computer control section provides control over all HP-21MX computer operations and data transfers by means of a microprogram stored in control memory, either ROM or Writable Control Store (WCS). Included in the control section is the 16 bit Instruction Register (IR), a 12 bit SAVE register, 12 bit ROM address register (RAR), 24 bit by 4096 word control store memory and the 24 bit ROM instruction register (RIR).

Control Store

The control store provides for microprogram storage in both non-volatile ROM and Writable Control Store (WCS) modules. The control store for the HP-21MX is configured into 16 modules (numbered 0 through 15) of 256 words each. A 12 bit address word is used for addressing control store. This means a total of 4096 words of addressable memory is available for storing microinstructions. Modules 0, 1 are reserved for holding the HP basic instruction set. Under normal circumstances these locations are not available for user microprograms. Generally modules 12 and 13 are assigned for user microprograms. The writable control store boards are usually assigned to module 12 and 13. All other module space is reserved for HP options. However if the HP options are not installed in the computer the user may use these module

locations by installing more writable control store boards or user burned PROMs. The control store word length is 24 bits wide as apposed to the 16 bit long words used for main memory instructions. When installed, modules 14 and 15, contain the microinstructions that interpret the floating point instructions and extended instruction group provided by HP.

Main Memory Section

The main program memory provides for storage of data and higher order languages like FORTRAN, ALGOL, etc. Included in this section are the M-register and the T-register. The normal machine instruction stored in Main Memory is 16 bits long. The M-register is equivalent to the Memory Address Register. The M-register holds a 15 bit address. This means up to 32K of main memory can be directly accessed through the M-register address. The M-register is loaded with an address from the S-bus. The T-register is a 16 bit buffer register connected to main memory and the S-bus. Main memory data words are transfered to and from the S-bus through the T-register.

Front Panel Section

The front panel section serves as a basic interface between the operator and the computer. Included in this section are two registers: the display register and the display indicator. Both registers are controlled by the base microinstruction routines. The display register is a set of six indicators which display the contents of the A, B, M, T, P or S registers.

Input/Output Section

The input/output section provides [illegible] for the computer to interface with external devices. Included in the I/O section is the

AD-A115 564 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOO--ETC F/G 9/2
MICROPROGRAMMING: A TOOL TO IMPROVE PROGRAM PERFORMANCE.(U)
DEC 81 J J STEIDLE
UNCLASSIFIED AFIT/GE/EE/81D-56 NL

2 of 2

END
DATE
FILMED
7 82
DTIC

central interrupt register (CIR), Interrupt contr. l (IAK), and I/O control and select logic decoding. The CIR is a 6 bit register. It contains the select code (address) of the interrupting device after and interrupt is recognized.

Software Support Tools for HP Microprogram Development

Microassembler: % MICRO

The microassembler is a Hewlett-Packard software utility program that processes Mneumonic microprograms and produces the binary 24 Bit word patterns (object program) that are to be loaded into WCS for controlling the computer.

Micro-Debug Editor: % MDEP

The Micro Debug Editor software allows the user to load, debug, and execute the object microprograms output from the microassembler into a Writable Control Store module.

Writable Control Software: %DVR36 & %WLOAD

%DVR36 is the driver software for the Writable Control Store. This software takes care of the data transfers through the I/O section and maintains conformity with the RTE operating system.

The WLOAD software permits loading the WCS with microprograms from a disk file or LU device.

APPENDIX E

The following listing shows the annotated contents of the answer file used in this study to generate the RTE-III system to support microprogramming.

```
YES                            *ECHO
40                             * #TRACKS IN OUTPUT FILE
SY2361,,10,                    * OUTPUT SYSTEM FILE NAME
7906                           * DISC MODEL
12                             * DISC SELECT CODE
800,0,0,2,0,22                 * SUBCHANNEL 0
256,0,2,2,0,6                  * SUBCHANNEL 1
536,132,2,2,0,22               * SUBCHANNEL 2
/E                             * TERMINATE SUBCHANNEL DEF
48                             * #128 WORD SECTGORS/TRK
1                              * SYSTEM SUBCHANNEL
NO                             * AUX DISC
11                             * TMG
0                              * PRIV INTERRUPT
NO                             * PRIV DIRVERS ACCESS COMM
YE                             * FG CORE LOCK
YE                             * BG CORE LOCK
50                             * SWAP DELAY
112                            * MEM SIZE
0                              * BOOT FILE
MAP MODULES                    * RELOACTE MODULES BY NAME
LINKS IN CURRENT               * CURRENT PAGE LINKING
*
*    RTE-III SYSTEM MODULES
*
REL,%CR3SY,,10,             * MEMORY RESIDENT SYSTEM
REL,%SYLIB,,10,             * RTE SYSTEM LIBRARY
REL,%CLIB,,10,              * RTE COMPILER LIBRARY
REL,%RLIB1,,10,             * RTE / DOS LIBRARY PARTS 1&2
REL,%RLIB2,,10,             * RTE/DOS LIBRARY PART 2
REL,%BMLIB,,10,             * RTE BATCH LIBRARY
REL,%#CMD3,,10,             * RTE-III COMMAND PROGRAM
REL,%BMPG1,,10,             * RTE BATCH MONITOR PROGRAM PART 1
REL,%BMPG2,,10,             * RTE BATCH MONITOR PROGRAM PART 2
REL,%BMPG3,,10,             * RTE BATCH MONITOR PROGRAM PART 3
REL,%DECAR,,10,             * RTE DECIMAL STRING ARITHMETIC
REL,%LDR3,,10,              * RTE III LOADR
REL,%MTM,,10,               * RTE MULTI-TERMINAL MONITOR
REL,%$PVMP,,10,             * $PVMP
REL,%FF4.N,,10,             * FORTRAN IV FORMATER
REL,%DVR00,,10,             * RTE TTY/PUNCH/PHOTO READER DVR
REL,%DVR36,,10,             * RTE WCS DRIVER
REL,%3DP43,,10,             * POWER FAILURE DRIVER
REL,%4DV05,,10,             * 26400 DRIVER
REL,%DVR32,,10,             * 7906 DISK DRIVER
[illegible]                 * [illegible]
[illegible]                 * [illegible]
REL,%EDITR,,10,             * EDITOR
```

```
REL,%XREF,,10,                    * CROSS-REFERENCE
REL,%ASMB,,10,                    * ASSEMBLER
REL,%SWTCH,,10,                   * SWITCH OPERATING SYSTEMS
REL,%LU,,10,
REL,%TT,,10,
REL,%PPONG,,10,
REL,%MICRO,,10,                   * RTE MICROASSEMBLER
REL,%MXREF,,10,                   * RTE MICRO CROSS ASSEMBLER
REL,%MLOAD,,10,                   * RTE MICRO LOAD UTILITY ROUTINE
/E


D.RTR,1,1                           * PARAMETERS
$$CMD,3,1
WHZAT,1,5
PPONG,1,32767
ASMB,3
XREF,3
LOADR,3
EDITR,3
AUTOR,3,1
PRMPT,3
R$PN$,3,50
/E                                  *TERMINATE
*EAU MACRO'S
.MPY,RP,100200
.DIV,RP,100400
.DLD,RP,104200
.DST,RP,104400
*
* FLOAT MACROS'S
*
.FAD,RP,105000
.FSB,RP,105020
.FMP,RP,105040
.FDV,RP,105060
IFIX,RP,105100
FLOAT,RP,105120
.MVW,RP,105777
2$DBL,RP,3
/E
25                                  * # BLANK ID SEGMENTS
15                                  * SHORT ID SEGMENTS
8                                   * MAX. # PARTITIONS
100                                 * FWA BP LINKAGE
40                                   * # OF IO CLASSES
0                                   * # LU MAPPING
25                                  * # RN
128,512                             * BUFFER LIMITS(LOW,HIGH)
```

```
* EQUIPMENT TABLE ENTRIES
*
12,DVR32,D                          * EQT #1
13,DVR05,B,X=13,T=13000             * EQT#2
10,DVR36                              * EQT #3
20,DVR00,B,T=13000                  *EQT #4
17,DVR01,B,T=500                     *EQT5
4,DVP43                              * EQT #6
/E
*
*
* DEVICE REFERENCE TABLE
*
2,0                                 * LU1 SYS. CONSOLE
1,1                                 * LU2 SYS. DISK(LOWER PLATTER)
0,0                                 * LU3 AUX. DISC
2,1                                 * LU4 L. CTU DRIVE
2,2                                 * LU5 R. CTU DRIVE
0,0                                 * LU6
0,0                                 * LU7
0,0                                 * LU8
0,0                                 * LU9


4,0                                 * LU10 TTY
5,0                                 * LU11 PAPER TAPE READER
0,0                                 * LU12
0,0                                 * LU13
0,0                                 * LU14
3,2                                 * LU15 WCS MODULE 12978A
0,0                                 * LU16
0,0                                 * LU17
0,0                                 * LU18
0,0                                 * LU19
1,0                                 * LU20 DISK SUBCHANNEL 0
1,2                                 * LU21 DISC SUBCHANNEL 2
0,0                                 * LU22
0,0                                 * LU23
0,0                                 * LU24
6,0                                 * LU25 POWER FAIL
/E
*
* INTERRUPT TABLE
*
4,ENT,$POWR
12,EQT,1
13,EQT,2
17,EQT,5
20,PRG,PRMPT
```

```
0
0
YES
YES
20
1,4,BG
2,7,BG
3,13,BG
4,14,BG
5,18,BG
6,18,BG
7,18,BG
/E
FMGR,12
EDITR,14
LOADR,14
ASMB,15
XREF,15
/E
/E
```

The following generation listing was produced by the On-Line Generator (RT3GN) for the RTE-III system generation done for the microprogramming configuration in this study.

```
ECHO?
TR,A92361
ECHO?
YES                                         *ECHO

EST. # TRACKS IN OUTPUT FILE?
40                                          * #TRACKS IN OUTPUT FILE

OUTPUT FILE NAME?
392361,,10,                                 * OUTPUT SYSTEM FILE NAME

TARGET DISK?
7906                                        * DISC MODEL

CONTROLLER CHNL?
12                                          * DISC SELECT CODE

# TRKS, FIRST CYL #, HEAD #,'# SURFACES, UNIT, # SPARES
  00?
800,0,0,2,0,22                              * SUBCHANNEL 0
  01?
256,0,2,2,0,6                               * SUBCHANNEL 1
  02?
536,132,2,2,0,22                            * SUBCHANNEL 2
  03?
/E                                          * TERMINATE SUBCHANNEL DEF

# 128 WORD SECTORS/TRACK?
48                                          * #128 WORD SECTGORS/TRK

SYSTEM SUBCHNL?
1                                           * SYSTEM SUBCHANNEL

AUX DISC (YES OR NO OR # TRKS)?
NO                                          * AUX DISC

TBG CHNL?
11                                          * TMG
```

```
PRIV. INT. CARD ADDR?
0                                      * PRIV INTERRUPT

PRIV. DRIVERS ACCESS COMMON?
NO                                     * PRIV DIRVERS ACCESS COMM

FG CORE LOCK?
YE                                     * FG CORE LOCK

BG CORE LOCK?
YE                                     * BG CORE LOCK

SWAP DELAY?
50                                     * SWAP DELAY

MEM SIZE?
112                                    * MEM SIZE

BOOT FILE NAME?




0                                      * BOOT FILE

PROG INPUT PHASE:
-
MAP MODULES                            * RELOACTE MODULES BY NAME
-
LINKS IN CURRENT                       * CURRENT PAGE LINKING
-
*
*   RTE-III SYSTEM MODULES
*
REL,%CR3SY,,10,                    * MEMORY RESIDENT SYSTEM
-
REL,%SYLIB,,10,                    * RTE SYSTEM LIBRARY
-
REL,%CLIB,,10,                     * RTE COMPILER LIBRARY
-
REL,%RLIB1,,10,                    * RTE / DOS LIBRARY PARTS 1&2
-
REL,%RLIB2,,10,                    * RTE/DOS LIBRARY PART 2
-
REL,%BMLIB,,10,                    * RTE BATCH LIBRARY
-
REL,%$CMD3,,10,                    * RTE-III COMMAND PROGRAM
-
REL,%BMPG1,,10,                    * RTE BATCH MONITOR PROGRAM PART 1
-
```

```
REL,%BMPG3,,10,                * RTE BATCH MONITOR PROGRAM PART 3
-
REL,%DECAR,,10,                * RTE DECIMAL STRING ARITHMETIC
-
REL,%LDR3,,10,                 * RTE III LOADR
-
REL,%MTM,,10,                  * RTE MULTI-TERMINAL MONITOR
-
REL,%$PVMP,,10,                * $PVMP
-
REL,%FF4.N,,10,                * FORTRAN IV FORMATER
-
REL,%DVR00,,10,                * RTE TTY/PUNCH/PHOTO READER DVR
-
REL,%DVR36,,10,                * RTE WCS DRIVER
-
REL,%3DP43,,10,                * POWER FAILURE DRIVER
-
REL,%4DV05,,10,                * 2648A DRIVER
-
REL,%DVR32,,10,                * 7906 DISK DRIVER
-
REL,%AUTOR,,10,                * AUTO RESTART
-
REL,%WHZT3,,10,                * WHZAT?
-
REL,%EDITR,,10,                * EDITOR
-
REL,%XREF,,10,                 * CROSS-REFERENCE
-
REL,%ASMB,,10,                 * ASSEMBLER



-
REL,%SWTCH,,10,                * SWITCH OPERATING SYSTEMS
-
REL,%LU,,10,
-
REL,%TT,,10,
-
REL,%PPONG,,10,
-
REL,%MICRO,,10,                * RTE MICROASSEMBLER
-
REL,%MXREF,,10,                * RTE MICRO CROSS ASSEMBLER
-
REL,%WLOAD,,10,                * RTE MICRO LOAD UTILITY ROUTINE
-
/E
```

```
NO UNDEFS

PARAMETERS
-
D.RTR,1,1                                * PARAMETERS
-
$$CMD,3,1
-
WHZAT,1,5
-
PRONG,1,32767
-
MSMB,3
-
XREF,3
-
LOADR,3
-
EDITR,3
-
AUTOR,3,1
-
ERMPT,3
-
R$PN$,3,50
-
/E                                       *TERMINATE

CHANGE ENTS?
-
*EAU MACRO'S
.MPY,RP,100200
-
.DIV,RP,100400
-
.DLD,RP,104200
-
.DST,RP,104400
-
*
* FLOAT MACROS'S
*
.FAD,RP,105000
-




.FSB,RP,105020
-
[illegible]

.FDV,RP,105060
```

```
-
IFIX,RP,105100
-
FLOAT,RP,105120
-
.MVW,RP,105777
-
.#DBL,RP,3
-
/E


# OF BLANK ID SEGMENTS?
25                                      * # BLANK ID SEGMENTS

# OF BLANK BG SEG. ID SEGMENTS?
15                                      * SHORT ID SEGMENTS

MAX NUMBER OF PARTITIONS?
8                                       * MAX. # PARTITIONS

FWA BP LINKAGE?
100                                     * FWA BP LINKAGE

SYSTEM

$OPSY(0099)02000 01777  92060-12003 REV.1740 770314

DISPM(0099)02021 04727  92060-16013 REV.1726 770527

RTIME(0099)05031 05604  92060-16014 REV.1710 770131

$ASCM(0099)05605 05677  92060-16015 REV.1631 760622

RTIOC(0099)05707 12413  92060-16016 REV.1740 770921

EXECD(0099)12446 14341  92060-16018 REV.1740 770614

$TRPN(0099)14351 14514  92060-16019 REV.A 750326

SCHED(0099)14546 20530  92060-16020 REV.1740 770614

$ALC (0099)20555 21002  92060-16017 REV.A 750505

$YCLB(0099)21003 21002  92001-16005 REV.1740 770613

$CLIB(0099)21003 21002  92060-12005 REV.1726 770523 $CLIB

RLIB1(0099)21003 21002   24998-16001  REV.1740  770812

RLIB2(0099)21003 21002   24998-16001  REV.1740  770812

[illegible]
```

```
$BMON<0099>21003 21002  92002-12001 REV.1631 760730
FF4.R<0099>21003 21002      24998-16002 REV.1726 770525
DVR00<0099>21017 22123  92001-50001 REV 1740 770808
DVR36<0099>22210 24221  RTE DVR36 13197-16001 REV.A 751221
DVP43<0099>24277 25110  92060-16001 REV.1633  760810
DVR05<0099>25174 30016      92001-16027  REV.1805 10-20-77
DVR32<0099>30020 31545  92060-16031 REV A 751024

*# OF I/O CLASSES?
40                                    * # OF IO CLASSES

*# OF LU MAPPINGS?
0                                     * # LU MAPPING

*# OF RESOURCE NUMBERS?
20                                    * # RN

BUFFER LIMITS (LOW, HIGH)?
128,512                               * BUFFER LIMITS(LOW,HIGH)

* EQUIPMENT TABLE ENTRY

EQT 01?
*
* EQUIPMENT TABLE ENTRIES
*
12,DVR32,D                            * EQT #1

EQT 02?
17 DVR05,B,X=13,T=18000               * EQT#2

EQT 03?
10,DVR36                              * EQT #3

EQT 04?
20 DVR00,B,T=18000                    *EQT #4

EQT 05?
17,DVR01,B,T=500                      *EQT5

EQT 06?
4,DVP43                               * EQT #6

EQT 07?
/E
```

```
* DEVICE REFERENCE TABLE

 1 = EQT #?
*
*
* DEVICE REFERENCE TABLE

*
2,0                                    * LU1 SYS. CONSOLE

 2 = EQT #?
1,1                                    * LU2 SYS. DISK(LOWER PLATTER)

 3 = EQT #?
0,0                                    * LU3 AUX. DISK

 4 = EQT #?
2,1                                    * LU4 L. CTU DRIVE

 5 = EQT #?
2,2                                    * LU5 R. CTU DRIVE

 6 = EQT #?
0,0                                    * LU6

 7 = EQT #?
0,0                                    * LU7

 8 = EQT #?
0,0                                    * LU8

 9 = EQT #?
0,0                                    * LU9

10 = EQT #?
4,0                                    * LU10 TTY

11 = EQT #?
[illegible]                            * LU11 PAPER TAPE READER

12 = EQT #?
0,0                                    * LU12

13 = EQT #?
0,0                                    * LU13

14 = EQT #?
[illegible]                            [illegible]
```

```
15 = EQT #?
3,2                                     * LU15 WCS MODULE 12978A

16 = EQT #?
0,0                                     * LU16

17 = EQT #?
0,0                                     * LU17

18 = EQT #?
0,0                                     * LU18

19 = EQT #?
0,0                                     * LU19

20 = EQT #?
1,0                                     * LU20 DISK SUBCHANNEL 0

21 = EQT #?
1,2                                     * LU21 DISC SUBCHANNEL 2

22 = EQT #?
0,0                                     * LU22

23 = EQT #?
0,0                                     * LU23

24 = EQT #?
0,0                                     * LU24

25 = EQT #?
6,0                                     * LU25 POWER FAIL

26 = EQT #?
/E

* INTERRUPT TABLE
-
*
* INTERRUPT TABLE
*
4,ENT,$POWR
-
12,EQT,1
-
13,[illegible]
```

```
17,EQT,5
-
20,PRG,PRMPT
-
/E

BP LINKAGE 01175

LIBRARY

   FRTN      35636 35741  92001-16005  761122
   TMVAL     35742 35761   92001-16005 741120
   .ENTR     35763 36052  750701   24998-16001


SUBSYSTEM GLOBAL MODULES

(NONE)


RT COMMON  00000
 CHANGE RT COMMON ?
0
RT COM      36054


BG COMMON  00000
 CHANGE BG COMMON ?





0
BG COM      36054


LWA BG COMMON  36053
ALIGN AT NEXT PAGE?
YES
LWA BG COMMON  37777

MEMORY RESIDENTS

D.RTR(0001)40001 42011  92002-16007 760528
   P.PAS     42022 42050  92002-16006 740801

WHZAT(0005)42051 43755  92060-16006 REV.1726 770520

LU    (0010)43760 44150

[illegible]
```

```
PRONG(2767)44347 44575


RT DISC RESIDENTS

(NONE)


BG DISC RESIDENTS

$$CMD(0001)40002 41334   92060-16076 REV 1710 770216
  RMPAR      41335 41372   770812  24998-16001

FMGR (0090)40002 40757   92002-16008 REV 1631 760627
  FM.CM      40760 42671   92002-16008 760616
  .DRCT      42776 43004   92001-16005 741120
  IFBRK      43005 43026   92001-16005 741120
  .DFER      43027 43100   770701  24998-16001
  OPEN       43101 43266   92002-16006 741205
  RMPAR      43267 43324   770812  24998-16001
  CLOSE      43325 43433   92002-16006  740801
  $OPEN      43434 43642   92002-16006   740801
  RWND$      43643 43753   92002-16006  740801
  R/W$       43756 44111   92002-16006  740801

FMGR0(0099)44113 44120   92002-16008  740801
  FK..       44121 45544
  COR.A      45545 45560   92001-16005 741120
  READF      45576 46333   92002-16006 760702
  REIO       46353 46455   92001-16005 741120
  RWNDF      46456 46537   92002-16006  740801
  F.PAS      46540 46566   92002-16006 740801
  RW$UB      46567 47040   92002-16006 750422
  LOCK.      47041 47102   92002-16006 760616
  FM.UT      47110 50250   92002-16006 760616
  CR..       50311 51373   92002-16008 760616
  NAM..      51374 51470   92002-16006 740801
  CREA.      51471 51542
  CREAT      51547 52024   92002-16006   741022




FMGR1(0099)44113 44241   92302-16008   760518
  .PARS      44242 45525   92002-16008 763025
  CNUMD      45526 45545   92001-16005 741120
  [illegible]
  CA..       45720 46141   92002-16008   760513
  [illegible]
  [illegible]
```

```
REIO      46762 47064   92001-16005 741120
P.PAS     47065 47113   92002-16006 740801
RW$UB     47114 47365   92002-16006 750422
EE..      47366 47426   92002-16008  760512
TR..      47427 47660   92002-16008  760614
MR..      47664 50126   92002-16008 760621
MESSS     50140 50367   92001-16005 770813
WRLG.     50370 50537   92002-16006 760622
CK.SM     50540 50653
SE..      50654 51040
IF..      51041 51251   92002-16008 740801
POSNT     51252 51515   92002-16006 760702
AB..      51516 51741   92002-16008 760517
DP..      51742 52007   92002-16008  760511

FMGR2(0099)44113 44123  92002-16008 760622
 IN.IT    44124 45116   92002-16006 760722
 IPUT     45117 45137   92002-16006    740801
 FID.     45140 45257
 NAM..    45260 45354   92002-16006 740801
 FM.UT    45355 46515   92002-16006 760616
 IN..     46546 50427   92002-16008  760527
 J.PUT    50505 50531   92002-16006  740801
 MSC.     50532 50566
 LOCK.    50567 50630   92002-16006 760616
 MC..     50631 51147   92002-16006  760511
 RC..     51150 51335
 PU..     51336 51560
 PURGE    51561 51657   92002-16006  740801

FMGR3(0099)44113 44120  92002-16008 760720
 DL..     44121 45372   92002-16002 760719
 READF    45416 46153   92002-16006 760702
 REIO     46176 46300   92001-16005 741120
 LOCF     46301 46467   92002-16006 750416
 P.PAS    46470 46516   92002-16006 740801
 RW$UB    46517 46770   92002-16006 750422
 MSC.     46771 47025
 FM.UT    47030 50170   92002-16006 760616
 F.SET    50217 50407   92002-16006 760719
 CS..     50410 50636   92002-16008 760518
 LULU.    50637 50727   92002-16006 760227

FMGR4(0099)44113 44122  92002-16008 [illegible]
 ST.DU    44123 45376   92002-16008  760622
 READF    45423 46160   92002-16006 760702
 REIO     46202 46304   92001-16005 741120
 RWNDF    46305 46366   92002-16006  740801
 LOCF     46367 46555   92002-16006 750416
 P.PAS    46556 46604   92002-16006 740801
 RW$UB    46605 [illegible]  92002-16006 750422
 CPOS.    47057 47150
```

CREAT	47131	47406	92002-16006	741022
NAM..	47407	47503	92002-16006	740801
CK.SM	47504	47617		
CO..	47620	50322		
FM.UT	50405	51545	92002-16006	760616
F.UTM	51546	52007	92002-16008	760514
FMGRC(0099)	44113	44125	92002-16008	760622
CNT.	44126	44362	92002-16006	760520
FCONT	44363	44460	92002-16006	751104
RU..	44461	45163	92002-16008	760530
FEADF	45164	45730	92002-16006	760702
REIO	45731	46033	92001-16005	741120
F.PAS	46035	46063	92002-16006	740801
RW$UB	46064	46335	92002-16006	750422
BUMP.	46336	46374	92002-16006	741025
SET.T	46375	46423	92002-16006	740801
TL.	46424	46457	92002-16006	760322
RP..	46460	47672	92002-16008	760514
MESSS	47703	50127	92001-16005	770813
LOCF	50135	50323	92002-16006	750416
J.PUT	50324	50350	92002-16006	740801
IPUT	50351	50371	92002-16006	740801
ID.A	50372	50461		
OPMES	50462	50652	92002-16008	760513
TL..	50653	50672		
ST.TM	50673	50727	92002-16006	741223
FMGR6(0099)	44113	44123	92002-16003	740801
CH..	44124	44164		
NAMF	44165	44336	92002-16006	740801
NAM..	44337	44433	92002-16006	740801
JO..	44434	45436	92002-16006	[illegible]
RNRQ	45437	45664	92001-16005	741120
$ALRN	45665	45772	92001-16005	741106
KCVT	45773	46005	92001-16005	741120
READF	46007	46544	92002-16006	760702
REIO	46545	46647	92001-16005	741120
POST	46650	46676	92002-16006	740801
F.PAS	46677	46725	92002-16006	740801
RW$UB	46726	47177	92002-16006	750422
SPOPN	47200	47250	92002-16006	741025
SET.T	47251	47277	92002-16006	740801
ST.TM	47300	47334	92002-16006	741223
R.FLG	47335	47403	92002-16006	741118
LULU.	47404	47474	92002-16006	760227
RANGE	47475	47520	92002-16006	740801
UNOFF	47535	50100	92002-16006	750128
AVAIL	50101	50173	92002-16006	741231
FO..	50174	51000	92002-16008	760616
[illegible]	51001	[illegible]	[illegible]	[illegible]
EX.TM	51226	51434	92002-16006	760627
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

```
  F.PAS     46743 46773  92002-16006 740801
  RW$UB     46774 47245  92002-16006 750422
  CL..      47246 47527
  LU..      47531 50652  92002-16008 760702
  RNRQ      50751 51176  92001-16005 741120
  $CLRN     51177 51304  92001-16005  741106
  KCVT      51305 51317  92001-16005 741120
  POST      51320 51346  92002-16006  740801
  SPOPN     51347 51417  92002-16006  741025
  LULU.     51420 51510  92002-16006 760227
  RANGE     51511 51534  92002-16006 740801
  AVAIL     51535 51627  92002-16006 741231

LOADR(0090)40002 51310  92060-16004 REV.1732 770811

PRMPT(0010)40002 40112  92001-16003 REV.B  741216
  EQLU      40113 40170  92001-16005 741120



R$PN$(0050)40002 40150  92001-16003 REV.B  741002
  EQLU      40151 40226  92001-16005 741120
  MESSS     40227 40453  92001-16005 770813

AUTOR(0001)40002 40440
  CLRIO     40441 40447  750701  24998-16001
  IAND      40450 40457  750701  24998-16001
  PAUSE     40460 40623  750701  24998-16001
  REIO      40624 40726  92001-16005 741120
  PAU.E     40727 40727  750701  24998-16001
  .OPSY     40730 40767  750701  24998-16001
  .DFER     40770 41041  750701  24998-16001
  FMTIO     41046 42475  24998-16002 REV.1715 770422 0800
  DBLE      42540 42571  750701  24998-16001
  SNGL      42572 42702  750701  24998-16001
  .FLUN     42703 42720  750701  24998-16001
  FMT.E.    42721 42761  24998-16002 REV 1610  760701
  FRMTR     42762 45563  24998-16002 REV 1610 760701
  .XPAK     45775 46160  750701  24998-16001

EDITR(0050)40002 44456  92002-16010 REV.C  750505
  REIO      [illegible]  [illegible] 741120
  .DFER     44562 44633  750701  24998-16001
  CREAT     44634 45111  92002-16006  741022
  RMPAR     45112 45147  750701  24998-16001
  OPEN      45150 45335  92002-16006 741203
  READF     45255 46112  92002-16006 760702
  CLOSE     46126 46274  92002-16006  740801
  [illegible]
  $OPEN     46332 46540  92002-16006  740801
  [illegible]
  [illegible]
```

```
  RWND$       47042 47152  92002-16006   740801
  R/W$        47153 47306  92002-16006   740801

XREF (0099)40002 46170  92060-16025 REV.A 750420
  .OPSY       46171 46230  750701   24998-16001

ASMB (0099)40002 45601  92060-16022 REV.B 760924

ASMBD(0099)45602 46410  92060-16023 REV.A 750420

ASMB1(0099)45602 47632  92060-16024 REV.A 750420

ASMB2(0099)45602 50133  92060-16025 REV.B 760924

ASMB3(0099)45602 46673  92060-16026 REV.A 750602

ASMB4(0099)45602 47347  92060-16027 REV.B 760924

SWTCH(0010)40002 55673  92060-16038 REV.1805 771213
  CNUMD       55674 55713  92001-16005 741120
  GETST       55722 56216  92001-16005 770208
  RMPAR       56223 56260  770812   24998-16001
  .DFER       56261 56332  750701   24998-16001
  OPEN        56333 56520  92002-16006 741205
  READF       56521 57256  92002-16006 760702
  REIO        57257 57361  92001-16005 741120
  LOCF        57362 57550  92002-16006 750416
  CLOSE       57551 57657  92002-16006   740801


  #OPEN       57672 60100  92002-16006    740801
  P.PAS       60102 60130  92002-16006 740801
  RW$UB       60131 60402  92002-16006 750422
  RWND$       60403 60513  92002-16006   740801
  R/W$        60514 60647  92002-16006   740801

DSEG0(0011)60650 61604  92060-16038 760715

DSEG5(0011)60650 62107  92060-16038 760715

MICRO(0099)40002 50263  RTE MICRO 92061-16001 REV.A 760818
  %WRIS       50264 50661  750701   24998-16001
  RMPAR       50662 50717  770812   24998-16001
  SREAD       50720 51362  750701   24998-16001
  .OPSY       51363 51422  750701   24998-16001
  CREAT       51423 51700  92002-16006   741022
  PURGE       51701 [illegible]  92002-16006   740801
  OPEN        52000 52165  92002-16006 741205
  [illegible]
```

```
REIO       52724 53026  92001-16005 741120
 .DFER     53027 53100  750701   24998-16001
 CLOSE     53101 53207  92002-16006   740801
 NAM..     53210 53304  92002-16006 740801
 $OPEN     53305 53513  92002-16006    740801
 P.PAS     53514 53542  92002-16006 740801
 RW$UB     53546 54017  92002-16006 750422
 RWND$     54026 54136  92002-16006   740801
 R/W$      54137 54272  92002-16006   740801

MXREF( 0099 40002 42707  92061-16002 REV.1913 771212
  RMPAR      42710 42745  770612   24998-16001
  SREAD      42746 43410  750701   24998-16001
  .OPSY      43411 43450  750701   24998-16001


RT PARTITION REQMTS:
(NONE)

BG PARTITION REQMTS:
  $$CMD  02 PAGES
  FMGR   07 PAGES
  LOADR  06 PAGES
  PRMPT  02 PAGES
  R$PN$  02 PAGES
  AUTOR  05 PAGES
  EDITR  05 PAGES
  XREF   05 PAGES
  ASMB   06 PAGES
  SWTCH  11 PAGES
  MICRO  08 PAGES
  MXREF  03 PAGES

LARGEST ADDRESSABLE PARTITION:
W/O COM 17 PAGES
W/  COM 17 PAGES


FWA MEM RESIDENT PROG AREA 445[illegible]
ALIGN AT NEXT PAGE?
YES




LWA MEM RESIDENT PROG AREA 45777

SYS AV MEM:  01024 WORDS

1ST DSK PG 00020
```

```
SYS AV MEM:  01024 WORDS

PAGES REMAINING: 00092
DEFINE PARTITIONS
-
1,4,BG
-
2,7,BG
-
3,13,BG
-
4,14,BG
-
5,18,BG
-
6,18,BG
-
7,18,BG
-
/E

MODIFY PROGRAM PAGE REQUIREMENTS?
-
FMGR,12
-
EDITR,14
-
LOADR,14
-
ASMB,15
-
XREF,15
-
/E

ASSIGN PROGRAM PARTITIONS?
-
/E
SYSTEM STORED ON DISC
SYS SIZE: 25 TRKS, 001 SECS(10)

RT3GN FINISHED
```

APPENDIX F

List of Commands used for backing up System on Disk

It is assumed the !DISKUP program is on a minicartridge. The following is a list of steps required to load and run !DISKUP for the microprogramming system configuration developed for this study.

1. To load !DISKUP from the minicartridge: insert the cartridge into the graphic terminal and set the computers S-register to 04302 octal. Press Preset/IBL. Press RUN.

Program should now load into computer and then HALT with 102077 octal displayed on the S-register display.

2. After the program is loaded, store an octal 13 into the S-register and set the P-register to an Octal 2. Press PRESET/RUN. The program will now execute interactively through the graphics terminal. Questions will be presented on the graphics terminal. You must supply the answers.

3. The following is a list of questions and the answers used in this study to backup the operating RTE-III system and other system files. It requires running two passes of !DISKUP. The first pass copies the operating system from cartidge 1 to the removable disk cartrige. The second pass copies the system files on cartridge 10 to the removable disk cartridge.

QUESTIONS	ANSWERS: 1st pass	2nd pass
TASK?	CO	CO
SOURCE DISK CHANNEL #?	12	12
SOURCE DISK TYPE?	7906	7906
SOURCE DISK DRIVE?	0	0
TYPE OF COPY?	FR	FR
RTE or DOS DISK?	RT	RT
FROM CYLINDER #?	0	0
# of TRACKS?	256	256
# of SURFACES?	2	2
STARTING HEAD?	2	2
DEST. DISK HEAD?	0	0
TO CYLINDER #?	0	132
# SURFACES?	2	2
STARTING HEAD #?	0	0
6144 WORD BUFFER DESIRED?	YES	YES
VERIFY*	YES	YES

* At this point hit any key for second pass: !DISKUP will respond with the second set of questions.

APPENDIX G

Discussions of Implementation Problems Incountered During Study

Many minor operating problems which will not be described here were encountered during the implimentation and microprogram development phases of this study. In the most part these operating problems were due to a lack of adequate and readily available HP documentation. Once the proper HP documentation and operating procedures were located many initial operating problems were resolved. Technical discussions with other HP users on base also helped tremendously in getting started and in identifying appropriate HP documentation.

In addition to all the operating problems encountered four major implementation problems occured. The following four events are presented as an example of the implementations problems encountered. The corrective action taken is listed where applicable.

First the base instruction set module proved to be a major initial problem with booting the computer up with the RTE operating system. The computer would not boot up properly. Usually several attempts were required to make the computer boot-up correctly. After boot-up the system acted highly unstable generally resulting in the computer halting. The problem was traced to the base instruction set board. The instruction board initially installed in the CDU was an older version. The solution was to replace this board with a later version board available in a second CPU unit at AFIT. Using the new base instruction board cleared up the boot-up problem.

A second problem occurred shortly after replacing the base instruction set; the main power [illegible] out. Fortuna-

ately the solution was to use the second CPU power supply unit available at AFIT while a new power supply was ordered from HP. However, this meant disassembling and reassembling the entire computer to change the main frame.

The third problem occurred with the papertape photo reader. Initially much of the software available at AFIT was still on papertapes. However the papertape reader was difficult to adjust properly. It would only work with black tapes. Keeping the optics clean helped eliminate some problems of reading the tapes. A better solution was initiated by installing a leader ROM to read magnetic tape cartridges from the systems HP 2648 graphics terminal. Transferring needed programs to magnetic tape cartridges solved the problem of reading data and programs from punched tapes.

The fourth problem was how to obtain a hardcopy printout of a program. This was partly solved by interfacing a TI TTY printer to the computer. However the unit would only operate at 110 band. Initial attempts to make it operate at 300 band failed. It was later learned the crystal on the interface board was not correct. Obtaining the proper crystal solved the problem and the TTY printer then worked with either 110 or 300 band.

APPENDIX H

How to Run ACTV Program on RTE-IVB System

User instructions:

The FORTRAN Program ACTV must be compiled and loaded before it can be run. Also the assembly subroutine RCORE must be assembled and loaded with ACTV. Compiling or assembling a program create binary files of the programs. It is assumed the source for ACTV is on file &ACTV and the source for RCORE is on file &RCORE. If ACTV must be compiled this can be cone by typing:

```
:RU, FTN4, &ACTV,, %ACTV
```

If RCORE must be assembled this can be done by typing:

```
:RU, ASMB, &RCORE,, %RCORE
```

Once ACTV is compiled and RCORE is assembled the programs must be loaded. This is done by typing:

```
:RU, LOADR
     /LOADR:   RE, %ACTV
               RE, %RCORE
               /E
```

After loading the programs you must set the program priority to be less than the file manager system program (FMGR). Usually the priority of FMGR is set at 90. You must set ACTV to a priority of 89 or less. The priority of ACTV is set by typing on the system console:

```
:PR, ACTV, 89
```

Before you can run the ACTV program you must also load the program you wish to monitor. Then assuming the program you are monitoring has

been loaded ACTV can be run by typing:

:RU, ACTV

Once running, ACTV will ask:

ACTIVITY PROFILE GENERATOR

TYPE PROGRAM NAME:

You must type in the name of the program you are monitoring at this point. Then the program will ask:

TYPE BOUNDS OF ACTIVITY PROFILE, LOWER-UPPER

XXXXX XXXXX X

You must now type in the address range in the monitored program you wish to examine. The address range is obtained from the load map of the monitored program. The load map is generated when the monitored program is loaded.

After entering the address ranges, hit any key. This will cause the computer to issue the command prompt. At this point type in:

:RU, (name of program being monitored)

The monitored program will now execute in its normal way. If the program is interactive you supply the data requested. When the program terminates, hit any key, The command mode will appear, then type the following:

:BR (Break)

The ACTV program will terminate and printout a histogram and cumulative plot of the activity monitored the address ranges you specified.

To zoom in on a program you must rerun ACTV with the address ranges narrowed in on the locations desired.

APPENDIX I

Sample Profile Analysis

This Appendix contains the activity profile analysis performed for the FOUR1 and FOURE FFT programs. The address load map for the two FFT programs is given in Table VI. The results of the activity profile generator program ACTV for the FOUR1 program is listed in Table VII. Figure 12 shows the FOUR1 results plotted. The zeros represent a histogram of the activity and the I symbols plot the cummulative distribution. Likewise Table VIII contains the activity results obtained for the FOURE program. Figure 13 is a plot of the FOURE activity results contained in Table VIII.

FOUR1 PROFILE ANALYSIS

Bit Reversal: The bit reversal software is computed between the address range 56363-56515 (Interval #3-12). The activity profile in Table VII shows 38 bits occured in this address range. This amounts to approximately 38/1044 or 3.64% activity for the bit reversal software.

Sine: The Sine values are computed in the address range 57023 - 57122 (Interval 35-41). The profile in Table VII shows 216 bits occured in this range out of a total of 1044 bits. Thus the sine computation takes approximately 216/1044 or 20.68% of the total activity shown in the profile.

Butterfly: The address range 56636 - 57023 (Interval 23-34) computes the butterfly values. As shown in the profile of Table VII 764 bits occured thus the butterfly for the FOUR1 program takes roughly 764/1044 or 73.18% of the total activity.

Overhead: All other activity amounted to 24 bits. This is 24/1044 or 2.29% of the total activity.

FOURE PROFILE ANALYSIS

Bit Reversal: The address range 55534-55656 (Interval 8-20) computes the bit reversal. The activity profile in Table VIII shows 102 hits. Hence the bit reversal activity is approximately 103/584 or 17.6%.

Sine: The address range 55706-55733 (Interval 24-26) computes the Sine values. The activity profile in Table VIII shows 79 hits occured in this range out of a total 584 hits. Thus the sine computation takes approximately 13.5% of the activity.

Butterfly: The activity in this section is roughly divided into two sections. First, the CONTINUE statements 110 and 120 in the DO loop for the butterfly computation show the following hits:

110 CONTINUE	56023-56032	95 hits
120 CONTINUE	56032-56041	13 hits
	TOTAL	108 hits

Hence the CONTINUE statements take 108-584 or 18.495 of the total activity.

Second, the address range 55742 to 55623 computes the following part of the butterfly code.

```
TEMP=W*DATA
DATA=DATA-TEMP
DATA-DATA+TEMP
```

The activity profile in Table VIII shows 260 hits occured in this address range. Hence 260/584 or 44.86% activity in this area.

The total butterfly therefore shows approximately 63.35% activity.

Overhead: The overhead is all the other address ranges which showed some activity. Lumping the activity of all the other sections resulted in 33 hits. Hence the overhead amounted to about 33/584 or 5.65%.

Table VI

Address Load Map for FOUR1 & FOURE FFT Programs

```
FFTC2   34042 55453
FOURE   55454 56203
RTIME   56204 56351
FOUR1   56352 57201
PAUSE   57202 57301   24998-1X253 REV.2101 801007
RMPAR   57302 57346   92068-1X025 REV.2101 800919
.DIO.   57347 57430   24998-1X331 REV.2101 800929
.EIO.   57431 60655   24998-1X329 REV.2101 801107
.FMCV   69656 63120   24998-1X333 REV.2101 800709
FMTIO   63121 64352   24998-1X328 REV.2101 800929
.IDER   64353 64466   24998-1X321 REV.2101 800731
.UFMP   64467 64501   24998-1X296 REV.2101 800731
REIO    64502 64626   92067-1X275 REV.2013 790316
PAU.E   64627 64627   24998-1X254 REV.2001 750701
PNAME   64630 64700   92068-1X035 REV.2101 800919
.CTOI   64701 65040   24998-1X035 REV.2013 791022
.ENTC   65041 65073   24998-1X155 REV.2001 750701
.CFER   65074 65135   24998-1X196 REV.2001 790523
.CMPY   65136 65240   24998-1X121 REV.2001 750701
.CDIV   65241 65377   24998-1X120 REV.2013 791227
.ITOI   65400 65516   24998-1X055 REV.2013 791017
..FCM   65517 65533   24998-1X182 REV.2001 750701
ERRO    65534 65623   24998-1X250 REV.2001 771122
.SNCS   65624 65765   24998-1X159 REV.2001 780424
CABS    65766 66061   24998-1X164 REV.2013 791016
ATAN    66062 66230   24998-1X177 REV.2001 780424
.CMRS   66231 66314   24988-1X171 REV.2001 780424
CMPLX   66315 66345   24998-1X138 REV.2101 800426
.CADD   66346 66406   24998-1X119 REV.2001 750701
.CSUB   66407 66447   24998-1X122 REV.2001 750701
ERO.E   66450 66450   24998-1X249 REV.2001 750701
.OPN?   66451 66474   24998-1X325 REV.2101 800803
SQRT    66475 66576   24998-1X181 REV.2001 780424
15 PAGES RELOCATED     15 PAGES REQ'D   NO PAGES EMA
LINKS:BP   PROGRAM:BG    LOAD:TE   COMMON:NC
 /LOADR:FFTC2 READY AT 1:59 PM FRI., 11 SEPT. 1981
 /LOADR:$END
```

Table VII

Actvitity Profile Results for FOUR1 FFT Program

FROM 056352 TO 057201 IN INCREMENTS OF 9

INTERVAL NO.	FROM	TO	NO OF HITS	NORMALIZED HITS	NORMAL ACCUM
1	000000	056352	805.	5.75000000	02679
2	056352	056363	0.	0.00000000	02679
3	056363	056374	3.	.02142857	02689
4	056374	056405	3.	.02142857	[illegible]
5	056405	056416	1.	.00714286	02703
6	056416	056427	6.	.04285714	02722
7	056427	056440	2.	.01428571	02729
8	056440	056451	2.	.01428571	02736
9	056451	056462	4.	.02857143	02749
10	056462	056473	5.	.03571428	02766
11	056473	056504	0.	0.00000000	02766
12	056504	056515	12.	08571428	02806
13	056515	056526	7.	.05000000	02829
14	056526	056537	8.	.05714285	02856
15	056537	056550	9.	.06428571	02886
16	056550	056561	0.	0.00000000	02886
17	056561	056572	0.	0.00000000	02886
18	056572	056603	0.	0.00000000	02886
19	056603	056614	1.	.00714286	02889
20	056614	056625	0.	0 00000000	02889
21	056625	056636	0.	0.00000000	02889
22	056636	056647	4.	.02857143	02902
23	056647	056660	17.	.12142856	02959
24	056660	056671	105.	.75000000	03308
25	056671	056702	42.	.30000001	03448

Table VII (cont.)

26	056702	056713	108.	.77142859	.03807
27	056713	056724	98.	.70000005	.04134
28	056724	056735	140.	1.00000000	.04600
29	056735	056746	29.	.20714286	.04696
30	056746	056757	55.	.39285713	.04879
31	056757	056770	13.	.09285714	.04922
32	056770	057001	71.	.50714290	.05159
33	057001	057012	38.	.27142859	.05285
34	057012	057023	44.	.31428570	.05432
35	057023	057034	55.	.39285713	.05615
36	057034	057045	41.	.29285717	.05751
37	057045	057056	18.	.12857142	.05811
38	057056	057067	17.	.12142856	.05868
39	057067	057100	19.	.13571429	.05931
40	057100	057111	25.	.17857143	.06014
41	057111	057122	41.	.29285717	.06151
42	057122	057133	1.	.00714286	.06154
43	057133	057144	0.	0.00000000	.06154
44	057144	057155	0.	0.00000000	.06154
45	057155	057166	0.	0.00000000	.06154
46	057166	057177	0.	0.00000000	.06154
47	057177	057210	0.	0.00000000	.06154
48	057210	057221	0.	0.00000000	.06154
49	057221	057232	0.	0.00000000	.06154
50	057232	057243	0.	0.00000000	.06154
51	057243	057254	0.	0.00000000	.06154
52	057254	057777	28197.	201.40714000	1.00000

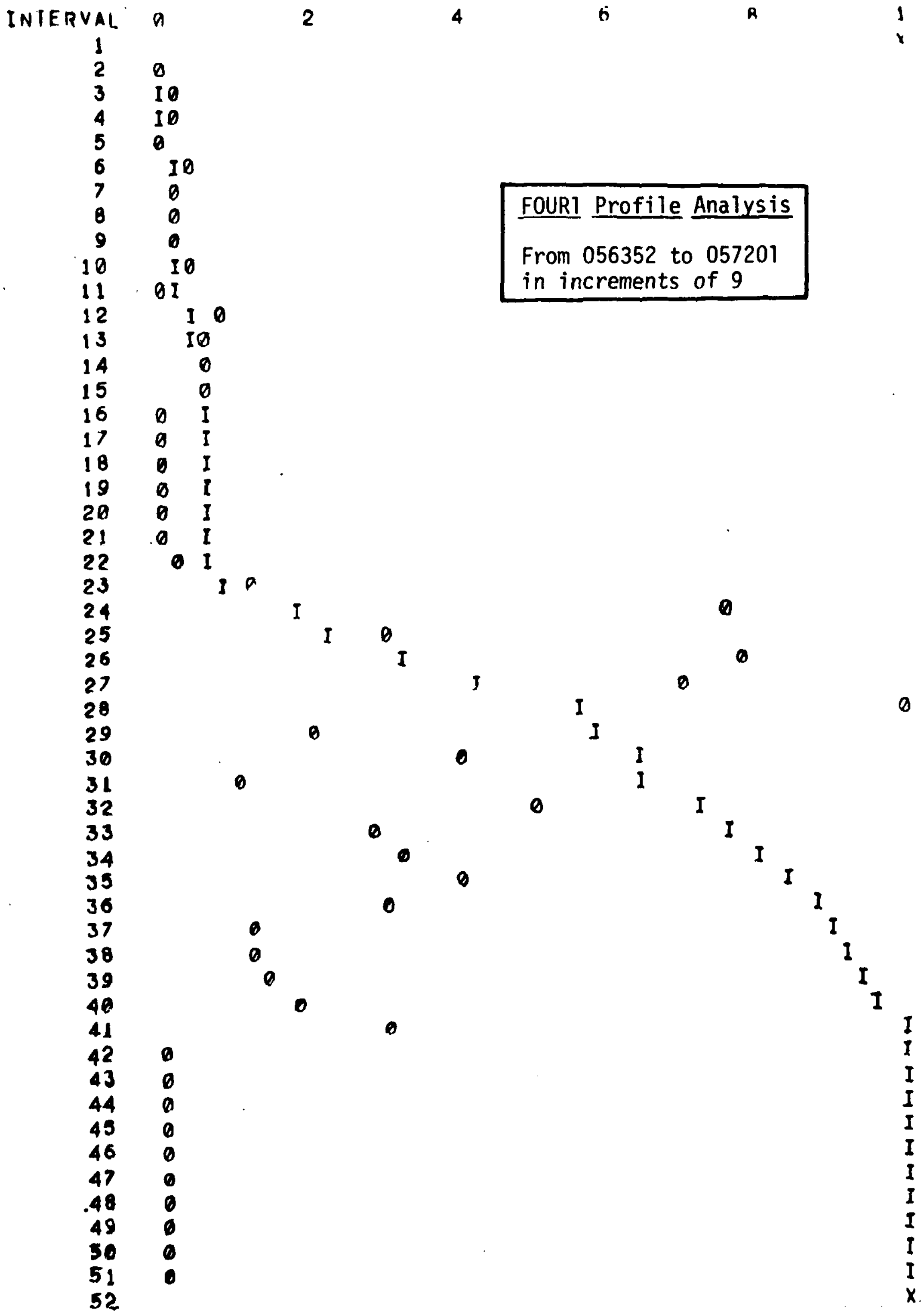


Figure 12. FOUR1 Activity Profile Plot

Table VIII

Activity Profile results for FOURE FFT Program

FROM 055454 TO 056203 IN INCREMENTS OF 7

INTERVAL NO.	FROM	TO	NO OF HITS	NORMALIZED HITS	NORMAL ACCUM
1	000000	055454	192.	2.02125280	.00643
2	055454	055463	0.	0.00000000	.00643
3	055463	055472	1.	.01052632	.00647
4	055472	055501	0.	0.00000000	.00647
5	055501	055510	0.	0.00000000	.00647
6	055510	055517	0.	0.00000000	.00647
7	055517	055526	0.	0.00000000	.00647
8	055526	055535	0.	0.00000000	.00647
9	055535	055544	0.	0.00000000	.00647
10	055544	055553	0.	0.00000000	.00647
11	055553	055562	5.	.05263158	.00664
12	055562	055571	6.	.06315790	.00684
13	055571	055600	1.	.01052632	.00687
14	055600	055607	2.	.02105263	.00694
15	055607	055616	4.	.04212526	.00707
16	055616	055625	5.	.05263158	.00724
17	055625	055634	36.	.37894738	.00845
18	055634	055643	7.	.07368422	.00868
19	055643	055652	3.	.03157895	.00878
20	055652	055661	30.	.31578946	.00979
21	055661	055670	3.	.03157895	.00989
22	055670	055677	0.	0.00000000	.00989
23	055677	055706	1.	.01052632	.00992
24	055706	055715	30.	.31578946	.01092
25	055715	055724	20.	.21052632	.01160

Table VIII (cont.)

26	055724	055733	29.	.30526316	.01257
27	055732	055742	0.	0.00000000	.01257
28	055742	055751	0.	0.00000000	.01257
29	055751	055760	40.	.42105263	.01391
30	055760	055767	19.	.19999999	.01454
31	055767	055776	73.	.76842105	.01699
32	055776	056005	35.	.36842108	.01816
33	056005	056014	78.	.82105267	.02078
34	056014	056023	15.	.15789473	.02120
35	056023	056032	95.	1.00000000	.02446
36	056032	056041	13.	.13684210	.02490
37	056041	056050	7.	.07368422	.02513
38	056050	056057	0.	0.00000000	.02513
39	056057	056066	5.	.05263158	.02530
40	056066	056075	2.	.02105263	.02537
41	056075	056104	16.	.16842106	.05590
42	056104	056113	3.	.03157895	.02601
43	056113	056122	0.	0.00000000	.02601
44	056122	056131	0.	0.00000000	.02601
45	056131	056140	0.	0.00000000	.02601
46	056140	056147	0.	0.00000000	.02601
47	056147	056156	0.	0.00000000	.02601
48	056156	056165	0.	0.00000000	.02601
49	056165	056174	0.	0.00000000	.02601
50	056174	056203	0.	0.00000000	.02601
51	056203	056212	0.	0.00000000	.02601
52	056212	077777	29064.	305.93683000	1.00000

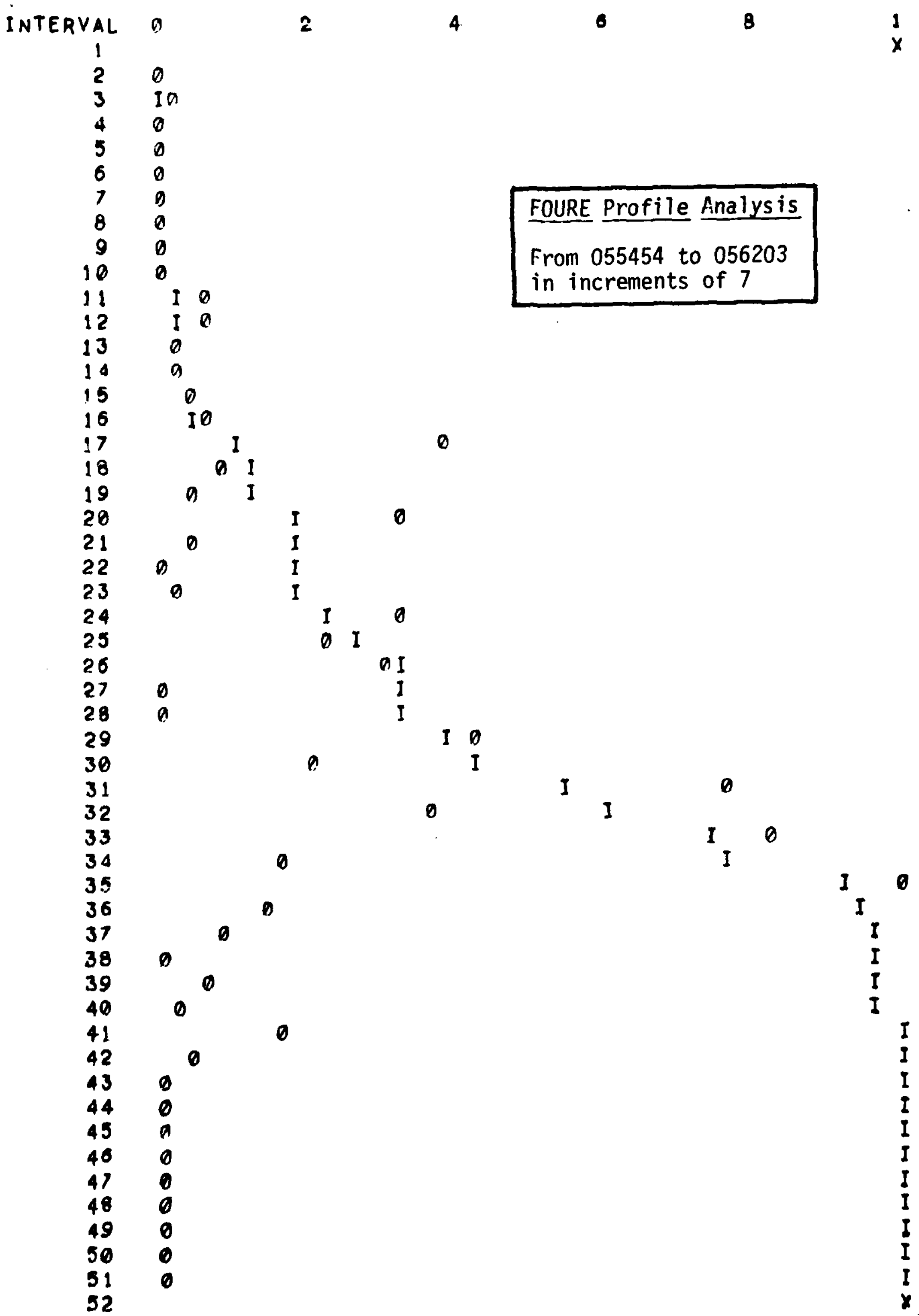


Figure 13. FOURE Activity Profile Plot

APPENDIX J

Appendix J contains a listing of the microprogrammed bit reversal sorting routine developed during this study. The bit reversal routine was named BINV. Also included in this Appendix is the Assembly interface routine that links the microprogram with the FORTRAN FFT calling program. A flow diagram showing the algorithm for the BINV program is also included in this Appendix in Figure 14.

```
MICMX,L,C
         ORG                   6005B
$CODE='MSBIN::10,REPLACE
*
*
*               THIS MICROPROGRAM PERFORMS THE BIT REVERSAL
*               AND EXCHANGES COMPLEX FLOATING POINT
*               DATA FOR COMPLEX RADIX 2 FFT PROGRAMS
*
*
*
*               THE CALLING SEQUENCE FROM FORTRAN IS:
*
*                    CALL BINV(N,NN,DATA)
*
*               WHERE  N ----- NUMBER OF COMPLEX DATA
*                      NN----- POWER OF 2 FOR FFT(N=2**NN)
*                      DATA--  COMPLEX DATA ARRAY (OR DATA
*                              ARRAY WITH ALTERNATING REAL
*                              AND IMAGINARY PARTS)
*
*
*
*               PARAMETERS N, NN, AND DATA ARE PASSED
*               TO THE MICROPROGRAM THROUGH AN ASSEMBLY
*               INTERFACE PROGRAM.
*
*                    CALLING SEQUENCE IS:
*
*                    JSB .ENTR
*                    DEF N
*                    DEF NN
*                    LDB N,I                B-REG CONTAINS N VALUE
*                    LDA NN,I               A-REG CONTAINS NN VALUE
*                    LDX DATA               X-REG CONTAINS STARTING
*                                           ADDRESS OF DATA ARRAY
*                    OCT 105605             INVOKES MICROPROGRAM
*
```

```
*********************************************************
*********************************************************
*
*               ALGORITHM:  BIT REVERSAL SORTING
*
*
*            THE ALGORITHM THIS MICROPROGRAM IMPLIMENTS
*       IS THE BIT REVERSAL, COMPARISON, AND COMPLEX FLOAT-
*       ING POINT DATA EXCHANGE FOR A RADIX 2 FAST FOURIER
*       TRANSFORM PROGRAM.  THE ALGORITHM IS AS FOLLOWS:
*
*
*       STEP 1.  FORM THE BIT REVERSED INDEX NUMBER IN
*                REGISTER S2. THE NUMBER TO BE BIT RE-
*                REVERSED IS IN REGISTER S4.
*
*
*
*
*       STEP 2.  COMPARE THE BINVERTED INDEX NUMBER S2
*                WITH DATA INDEX Y.  IF S2 IS GREATER
*                THEN Y THEN EXCHANGE THE COMPLEX DATA
*                "IN-PLACE".  FIRST FORM THE ADDRESSES
*                FOR THE DATA EXCHANGE AND THEN JUMP
*                TO THE SWITCH ROUTINE LABLE - SW.
*                THE DATA ADDRESS FOR THE INDEX Y IS
*                STORED IN THE A REGISTER.  THE DATA
*                ADDRESS FOR THE BIT REVERSED NUMBER
*                IS STORED IN THE B-REGISTER.
*
*
*       STEP 3.  IF S2 IS LESS THAN OR EQUAL TO Y THEN
*                DO THE NEXT INDEX NUMBER Y
*
*
*       STEP 4.  CHECK TO SEE IF ALL DATA POINTS HAVE
*                BEEN DONE.  CONTINUE STEPS 1-3 UNTIL
*                ALL DATA HAS BEEN PROCESSED.
*
*
*       STEP 5.  AFTER ALL DATA HAS BEEN EXCHANGED
*                THEN RESTORE VALUE OF P-REGISTER AND
*                THE M-REGISTER AND RETURN TO THE CALLING
*                PROGRAM.
*
*
*********************************************************
```

```
*****************************************************************
*
* BEGIN BIT REVERSAL
*
*
*****************************************************************
MBINV                   S11  P         SAVE THE RETURN ADDRESS
                   DEC  S12  B         SAVE N-1 IN S12
                        S10  A         SAVE NN IN S10
                   ZERO Y              INITIALIZE INDEX COUNTER
NEXTY                   S4   Y         PUT INDEX COUNT IN S4
                   ZERO S2             INITIALIZE BINVERT REGISTER
                        S5   S10       PUT NN COUNT IN S5
LOOP          L1        S2   S2         LEFT SHIFT S2 1 BIT
                        S4   S4         CHECK INDEX
         JMP  CNDX AL0  RJS  *+2        AL0 BIT--JMP NEXT INST
*                                      IF ZERO
                   INC  S2   S2        ADJUST BINVERT COUNT
              R1        S4   S4        RIGHT SHIFT INDEX
                   DEC  S5   S5        ADJUST BIT COUNT NN=NN-1
         JMP  CNDX TBZ       *+2       IS COUNT ZERO? YES,JMP INST.
         JMP                 LOOP      NO!!! NOT DONE YET!!!
*
*
*******************************************************************
*
*             NOW COMPARE BINVERTED INDEX WITH DATA INDEX
*
*
                        L    S2        PUT BINV # INDEX IN L-REG
                   SUB       Y         SUBTRACT INDEX# FROM BINV #
         JMP  CNDX AL15 RJS  CONTINUE  IF(BINV#)INDEX)
*                                      THEN EXCHANGE DATA
*
*******************************************************************
*
* NOW FORM THE ADDRESSES FOR THE DATA EXCHANGE
*
*
                        S5   Y
              L1        S5   S5        FORM Y INDEX*4
              L1        S5   S5
              L1        S2   S2        FORM BINV# INDEX*4
              L1        S2   S2
*
*
                        L    X         PUT DATA ADDRESS IN L REG
                   ADD  A    S5        A=ADDRESS DATA(INDEX)
                   ADD  B    S2        B=ADDRESS DATA(BINV#)
*
*******************************************************************
*
```

```
*NOW JUMP TO SWITCH ROUTINE
*
*
        JMP                  SW        SWITCH COMPLEX DATA
*
***********************************************************
*
CONTINUE                L    S12       NOW CHECK TO SEE
                  SUB        Y         IF ALL DATA POINTS
        JMP  CNDX TBZ        .DONE     ARE DONE
                  INC   Y    Y         NO!!!BUMP INDEX Y COUNT
        JMP                  NEXTY     AND DO NEXT INDEX
*
*
*
**********************************************************
*
.DONE                   P    S11       RESTORE P
        READ RTN  DEC   M    S11       AND M REGISTER AND
*                                      RETURN TO MAIN PROGRAM
*
*
**********************************************************
*
*
*
*            DATA EXCHANGING PROCEDURE
*
*
*
*            THE MAIN MICROPROGRAM PASSES:
*            ADDRESS OF DATA(INDEX) IN A REG.
*            ADDRESS OF DATA(BINV#) IN B REG.
*
*            LET INDEX = I
*            LET BINV# = J
*
*
*
*
****************************************************************
SW                      P    A         PUT ADDRESS DATA(I)IN P
        READ      INC   PNM  P         START A READ,P->M,P=P+1
                        S1   TAB       STORE REAL DATA(I) WORD1 IN S1
        READ      INC   PNM  P         START READ,P->M,P=P+1
                        S2   TAB       STORE REAL DATA(I) WORD2 IN S2
        READ      INC   PNM  P         START READ P->M,P=P+1
                        S3   TAB       STORE IMAG DATA(I) WORD3 IN S3
        READ      INC   PNM  P         START READ P->M,P=P+1
                        S4   TAB       STORE IMAG DATA(I) WORD4 IN S4
```

```
*
                      P     B          PUT DATA(J) ADDRESS IN P-REG
*
*
          READ   INC  PNM   P          READ DATA(J)WORD1,P->M,P=P+1
                      S5    TAB        STORE DATA(J) WORD1 IN S5
          READ   INC  PNM   P          START READ P->M, P=P+1
                      S6    TAB        STORE DATA(J) WORD2 IN S6
          READ   INC  PNM   P          START READ,P->M, P=P+1
                      S7    TAB        STORE DATA(J) WORD3 IN S7
          READ        M     P          START READ,P->M, P=P+1
                      S8    TAB        STORE DATA(J) WORD4 IN S8
*
*
*
*NOW PERFORM SWITCH ON DATA(I) AND DATA(J)
*
*
*
                 DEC  PNM   P          LAST P->M, P=P-1
          WRTE        T     S4         WRITE IMAG DATA(I) WORD4
                 DEC  PNM   P          START WRITE P->M,P=P-1
          WRTE        T     S3         WRITE IMAG DATA(I) WORD3
                 DEC  PNM   P          START WRITE, P->M,P=P-1
          WRTE        T     S2         STORE REAL DATA(I) WORD2
                 DEC  PNM   P          START WRITE P->M, P=P-1
          WRTE        T     S1         WRITE REAL DATA(I) WORD1
*
*
*NOW SWITCH DATA(J) WORDS
*BUT
*FIRST RESTORE PROPER STARTING ADDRESS
*
                      P     A          ADDRESS OF DATA(I)->P
*
*
                 INC  PNM   P          START WRITE
          WRTE        T     S5         STORE REAL DATA(J) WORD 1
                 INC  PNM   P          START WRITE
          WRTE        T     S6         STORE REAL DATA(J) WORD 2
                 INC  PNM   P          START WRITE
          WRTE        T     S7         STORE IMAG DATA(J) WORD 3
                 INC  PNM   P           START WRITE
          WRTE        T     S8         STORE IMAG DATA(J) WORD 4
*
*
*
*
*                 END OF EXCHANGE ROUTINE
*                 CONTINUE WITH NEXT INDEX
*
*
          JMP               CONTINUE
          END
```

```
ASMB,L,C
*
*
* THIS ASSEMBLY PROGRAM PASSES
* THE PRAMATERS N, NN, AND DATA
* TO A MICROCODED BIT REVERSAL
* ROUTINE. THE DATA ARRAY CONTAINS
* COMPLEX FLOATING POINT NUMBERS
*
* THE CALLING ROUTINE FROM FORTRAN
* IS:
*        CALL BINV(N, NN, DATA)
*
*
      NAM .BINV,7
*
      ENT BINV
      EXT .ENTR
N     BSS 1
NN    BSS 1
DATA  BSS 1
BINV  NOP
      JSB .ENTR
      DEF N
      DEF NN
*
*
*
      LDB N,I                        N VALUE
      LDA NN,I                       NN VALUE
      LDX DATA                       DATA STARTING ADDRESS
*
*
*     EXECUTE THE MICROCODE
*
*     MICROCODE STARTS AT
*     WCS ADDRESS 6005B
*
****************************************************
      OCT 105605
****************************************************
      JMP BINV,I
      END
```

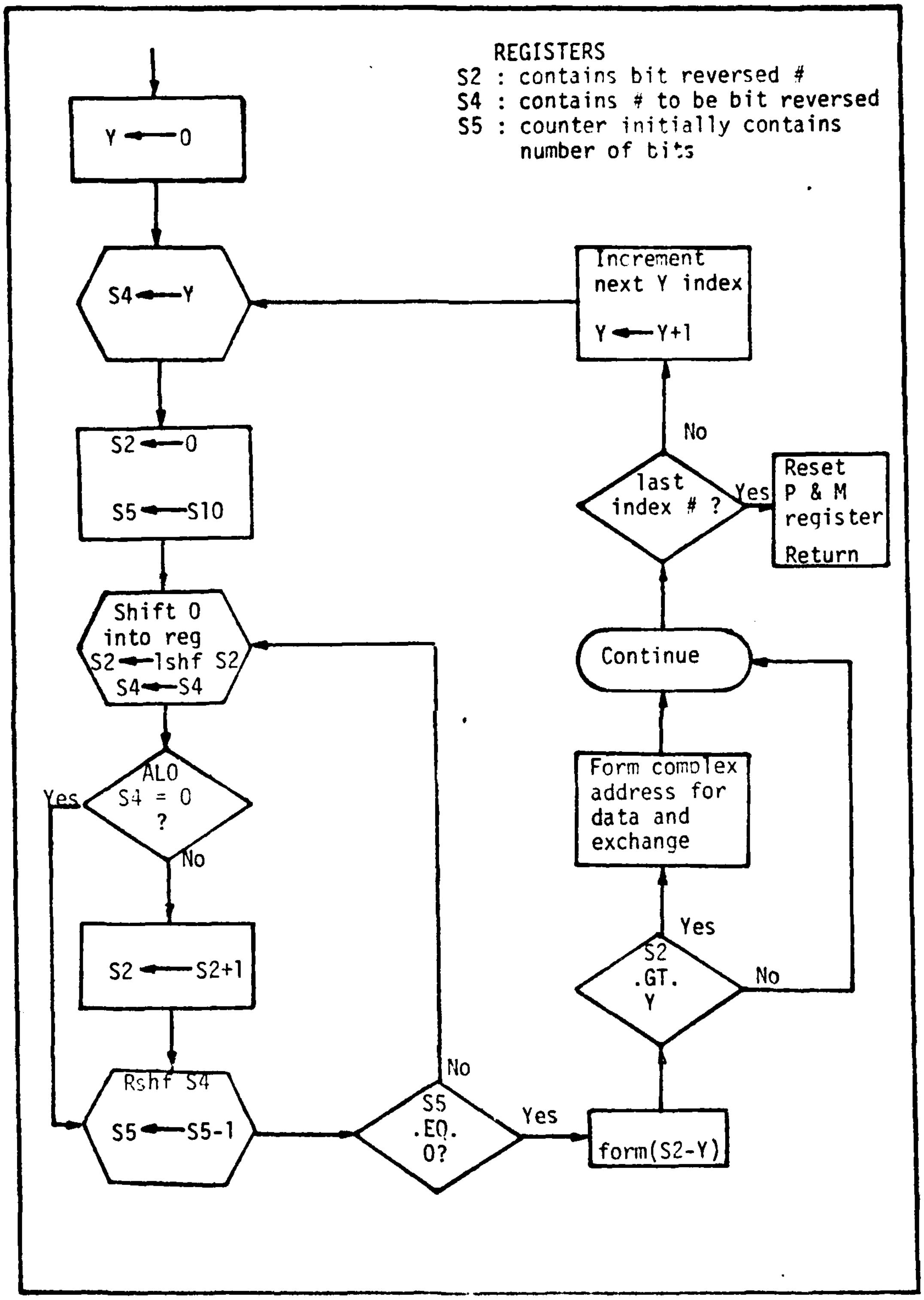


Figure 1.. Flow Diagram of [illegible]

APPENDIX K

Theoretical Calculation of Microprogrammed Bit Reversal Algorithm (BINV) Execution Time

Micrprograms tend to be short programs. The microprogrammed bit reversal programmed for this study was 70 statements long. The 21MX basic cycle time to execute one microinstruction is 325 nsec. long. It is therefore simple to calculate the run time for the microprogram developed by noting the number of statements and the number of times they are executed. This is done as follows:

1. To form the bit reversed number required 8 microinstructions. This code loops through $N(2^N)$ times. Therefore we have

$$\text{Time} = 2^N(N) \times 8 \times 325 \times 10^{-9} = 2.6 \times 10^{-6}\ N(2^N) \text{ sec.} \qquad (4)$$

2. To set up and initialize registers required 4 microinstructions which execute only once. Therefore the time for these microinstructions is:

$$\text{Time} = 3 \times 325 \times 10^{-9} = 1.3 \times 10^{-6} \text{ sec.} \qquad (5)$$

3. To form the data exchange addresses required 8 microinstructions. This code loops through NN times, where NN is the number of times the bit reversed index is greater than the number that was bit reversed. Thus the time is:

$$\text{Time} = 8 \times 325 \times 10^{-9}\ NN = 2.6 \times 10^{-6}\ NN \text{ sec.} \qquad (6)$$

4. To perform the complex data exchnage required 37 microinstructions. These microinstructions are executed NN times. Thus the time required to execute these instructions is:

$$\text{Time} = 37 \times 325 \times 10^{-9}\ NN = 12.025 \times 10^{-6}\ NN \text{ sec.} \qquad (7)$$

5. To perform the bit reversal compare required 3 microinstructions. These statements execute 2^N times. Hence the time required to execute this code is:

$$\text{Time} = 3 \times 325 \times 10^{-9} \times 2^N - 9.75 \times 10^{-7} \times 2^N \qquad (8)$$

6. To form the next index number required 8 microinstructions. They are executed 2^N times. Therefore the time to execute these instructions is:

$$\text{Time} = 8 \times 325 \times 10^{-9} \times 2^N = 2.6 \times 10^{-6} \times 2^N \qquad (9)$$

7. To reset the P and M registers and return to the calling program required 2 microinstructions. These instructions execute only once, after the routine is done. The time is:

$$\text{Time} = 8 \times 325 \times 10^{-9} = 0.65 \times 10^{-6} \text{ sec.} \qquad (10)$$

Combining the above executions times gives the total run time for the microcoded routine. Hence the run time is given by:

$$\text{Run Time} = (2.6N + 3.575)2^N + 14.6NN + 1.95 \text{ microseconds} \qquad (11)$$

Using Eq 11 the run time for the'microcoded bit reversal sorting routine can be theoretically calculated. The run times for various number of points of the FFT are shown below:

N	2^N	NN	Run Time (microseconds)
2	4	1	51.4
3	8	2	121
4	16	6	311
5	32	12	704
6	64	28	1630
7	128	56	3590
8	256	120	7994
9	512	240	17317

VITA

John J. Steidle was born on 17 June 1946 in Covington, Kentucky. He graduated from Covington Catholic High School in Park Hills, Kentucky in 1964. He attended the University of Detroit and received the degree of Bachelor of Electrical Engineering in 1969. In 1970 he graduated from Thomas More College, Covington, Kentucky and received an AB degree in Physics. He entered civil service at Wright-Patterson Air Force Base, Ohio in 1967 as a junior co-op student while at the University of Detroit. He was employed full-time at Wright-Patterson after graduation from the University of Detroit in 1969.

As project engineer for the Air Force he supported the Electronic Warfare System Program Office on numerous electronic warfare simulation programs involving the EDE, DEES, AF-EWES and REDCAP simulation facilities.

In October 1980 he entered the School of Engineering, Air Force Institute of Technology where he is working toward a masters degree in electrical engineering.

Permanent Address: 45 Winding Dr.
Enon, Ohio 45323

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: AFIT/GE/EE/81D-56
2. GOVT ACCESSION NO.: AD-A115 524
3. RECIPIENT'S CATALOG NUMBER:
4. TITLE (and Subtitle): MICROPROGRAMMING: A TOOL TO IMPROVE PROGRAM PERFORMANCE
5. TYPE OF REPORT & PERIOD COVERED: MS Thesis
6. PERFORMING ORG. REPORT NUMBER:
7. AUTHOR(s): John J. Steidle
8. CONTRACT OR GRANT NUMBER(s):
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Air Force Institute of Technology (AFIT-EN) Wright-Patterson Air Force Base, Ohio 45433
10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS:
11. CONTROLLING OFFICE NAME AND ADDRESS:
12. REPORT DATE: December 1981
13. NUMBER OF PAGES: 145
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office):
15. SECURITY CLASS. (of this report): UNCLASSIFIED
15a. DECLASSIFICATION/DOWNGRADING SCHEDULE:
16. DISTRIBUTION STATEMENT (of this Report): Approved for public release; distribution unlimited
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report):

15 APR 1982

18. SUPPLEMENTARY NOTES:
Approved for public release; IAW AFR 190-17
Fre~~deric C. Lynch, Maj~~, USAF
Director of Public Affairs

Dean for Research and Professional Development
Air Force Institute of Technology (ATC)
Wright-Patterson AFB, OH 45433

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Microprogramming	Bit Reversal
21MX Computer	Electronic Warfare
Fast Fourier Transform	

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A user microprogramming capability was implemented on AFIT's HP 21MX RTE-III computer system. The AFIT computer will be used to support student research in microprogramming projects involving real time digital signal processing and special time critical programs for military environments.

This study further looks at a specific microprogramming technique to tailor application programs for improving a program's execution time. A feasibility study to analyze program activity for microprogram improvements is done for the

fast Fourier transform. The Bit Reversal sorting routine is microcoded to demonstrate the technique. Response of the FFT programs is analyzed using an activity profile generator program, and the difference in execution speed of the programs with and without the microprogrammed bit reversal routine is measured and compared.

DATE

LMED

8